9-27-2019

An Approach Of Features Extraction And Heatmaps Generation Based Upon Cnns And 3D Object Models

Shivani Pachika
University of Windsor

AN APPROACH OF FEATURES EXTRACTION AND HEAT MAPS GENERATION BASED UPON CNNS AND 3D OBJECT MODELS

By

Shivani Pachika

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2019

AN APPROACH OF FEATURES EXTRACTION AND HEAT MAPS GENERATION BASED UPON CNNS AND 3D OBJECT MODELS

by

Shivani Pachika

APPROVED BY:

_____

M. Hlynka

Department of Mathematics and Statistics

_____

A. Ngom

School of Computer Science

_____

X. Yuan, Advisor

School of Computer Science

September 27th, 2019

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights. Any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have added copies of such copyright clearances to my appendix.

I also declare that this is an exact copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The rapid advancements in artificial intelligence have enabled recent progress of self-driving vehicles. However, the dependence on 3D object models and their annotations collected and owned by individual companies has become a major problem for the development of new algorithms. This thesis proposes an approach of directly using graphics models created from open-source datasets as the virtual representation of real-world objects. This approach uses Machine Learning techniques to extract 3D feature points and to create annotations from graphics models for the recognition of dynamic objects, such as cars, and for the verification of stationary and variable objects, such as buildings and trees. Moreover, it generates heat maps for the elimination of stationary/variable objects in real-time images before working on the recognition of dynamic objects. The proposed approach helps to bridge the gap between the virtual and physical worlds and to facilitate the development of new algorithms for self-driving vehicles.

DEDICATION

To the almighty God, my beloved parents Mr. Devender Reddy and Mrs. Jaya, my loving sister Miss Bhavishya, for their unconditional love, support, sacrifice, and encouragement.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my supervisor Dr. Xiaobu Yuan for his encouragement and support in presenting this thesis work. I have been extremely fortunate to have a supervisor who has given me the freedom to explore things on my own, and with his patient guidance whenever my steps faltered. His kindness and support have helped me overcome many problems, throughout my project. It has been a great pleasure to work under him as one of his research project students.

I would like to extend my gratitude to my thesis committee members Dr. Myron Hlynka (Department of Mathematics and Statistics) and Dr. Alioune Ngom (School of Computer Science) whose suggestions and recommendations significantly improved the quality of this work. I would also like to thank them for spending their valuable time to assess the thesis from my proposal to defense. I would also like to thank the secretaries of the Computer science department, Mrs. Christine Weisener and Mrs. Melissa Robinet for all the assistance, motivation and support given to me.

My special thanks goes to my friends and relatives for their constant encouragement and love they provided to me at all times. I express my sincere appreciation for my fellow lab mates for the invaluable support they provided during all stages of my thesis work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS/SYMBOLS

2D	2-dimensional
3D	3-dimensional
RGB	Red Green Blue
OSM	OpenStreetMap
XML	Extensible Markup Language
GIS	Geographic Information System
GPS	Global Positioning System
LIDAR	Light Detection and Ranging
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Feature
HOG	Histogram of Oriented Gradients
FAST	Features from Accelerated Segment Test
NMS	Non-Maximal Suppression
BoF	Bag-Of-Features
SVM	Support Vector Machine
SSD	Single-Shot Detector
ReLU	Rectified Linear Units
RoI	Region of Interest
CNN / ConvNet	Convolutional Neural Network
FCN	Fully Convolutional Network
R-CNN	Regions based/with Convolutional Neural Networks
RPN	Region Proposal Networks
R-FCN	Region-based Fully Convolutional Networks
YOLO	You Only Look Once
mAP	mean Average Precision

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

A self-driving or autonomous vehicle can detect its environment and navigate without human input. It can detect an environment using a range of methods, including cameras, GPS and computer vision [2]. Automated driving is a rapidly advancing application area with a complex structure and lots of progress in Deep Learning. Companies like Google, Uber, Tesla, Mercedes and BMW have already released or foraying quick. The examples of the self-driving vehicle are shown in Figure 1 and Figure 2.

According to recent outcomes of studies [36], feature extraction techniques are of critical significance to many pattern recognition applications and systems involving detection, recognition, registration, matching, reconstruction and classification. Object detection in real complex environments is a challenging task for autonomous driving in the aforementioned applications. A typical pipeline for object detection can be divided into three stages: the selection of informative regions, extraction of features and classification. Deep Learning has reformed Computer Vision and is the core innovation behind the capabilities of a self-driving vehicle [1]. Convolutional Neural Networks (CNNs) are pivotal to the improvement of object detection in this deep learning revolution.

We propose the integration of a new source of a priori information, the virtual 3D city model for self-driving cars for object detection, feature extraction and heatmap generation.

Figure 1: Google's self-driving car [69]

Figure 2: Uber's autonomous car [11]

1.2 Sensing Technology Comparison

Rating: H = High, M=Medium, L = Low	Camera	Radar	LiDAR
Object Detection	M	H	H
Classification	H	M	L
Close-Proximity Detection	M	H	L
Speed Detection	L	H	M
Lane Detection	H	L	L
Traffic Sign Recognition	H	L	L
Range	H (200m)	H (250m)	M (120m)
Work in Rain, Fog, Snow	L	H	M
Work in Low Light	L	H	H
Work in Bright Light	M	H	H
Size	Small	Small	Medium
Cost	$	$$	$$$$

Table 1: Sensing Technology Comparison [49]

In consideration of all these differences shown in Table 1, autonomous vehicles can use a camera because of its added advantages and with better machine vision, it can identify everything it detects and navigates accordingly.

1.3 Problem Statement

- In the real world, there are many associated risks and cost issues to acquire training data for self-driving artificial intelligence algorithms.
- The dependence on 3D object models and their annotations, collected and owned by individual companies is a hindrance to the development of the new algorithms.
- Their approaches remain fundamentally bounded by massive amounts of human-annotated training data.
- This time-consuming process impedes the progress of these deep learning efforts.

1.4 Motivation

In recent years, exploration in the area of self-driving cars has increased. Nowadays, outdoor positioning systems often rely on GPS because of its affordability and convenience. However, GPS suffers from the occurrence of satellite masks especially in urban environments, under bridges, tunnels or in forests. To provide continuous, accurate, and high integrity position data, satellite-based localization systems should incorporate additional sensors (as proprioceptive sensors or environment perception sensors) or database (for example 2D digital map). Nevertheless, using only incremental encoders placed on the rear wheels and gyroscopes is not sufficient in case of long GPS outages, because the Dead-Reckoning (DR) localization is prone to drift error due to accumulation of data. As an alternative, a new approach to back up the limitations of GPS and DR sensors based localization is required. The proposed approach must aim at providing absolute positioning information by integrating a virtual 3D city model and an on-board camera in the localization process. If a GPS measurement is available, then this data is used to update the prediction, else the 3D model/camera-based pose estimation corrects the prediction. We can determine the vehicle pose by registering a priori virtual 3D city model with a captured 2D image. These virtual environments can be purposefully and increasingly challenging for critical applications, and will also be able to train a self-driving car to drive in an area full of simulated people, or a robot to respond to complex challenges and

variances before being placed on a real assembly line. In addition to the above, there is still a visible gap between machine performance and that of humans', inspiring and recommending promising future bearings for the deployment of computer vision applications.

1.5 Thesis Contribution

- The target of ongoing research is to present an approach that directly uses graphic models from open source datasets as the virtual presentation of real-world objects to develop new computer vision tasks related to self-driving vehicles.
- To train the system in a way that they can not only detect objects but also differentiate with high accuracy.
- To train a network to extract features of 3D models for verification and elimination of static and variable objects and identification of dynamic objects.
- Sensor and object recognition technologies for self-driving cars must fulfill enhanced requirements in terms of accuracy, unambiguousness, robustness, space demand and of course, costs.

1.6 Structure of the thesis

The overall structure of the thesis is organized in the following way: Chapter 2 begins with the background study for feature extraction and object detection techniques. In Chapter 3, literature survey and related works about feature extraction and object detection techniques are discussed extensively. The proposed system is introduced in Chapter 4: the feature extraction of cars and buildings, details of the overall system and connection of this thesis work with the overall system are also discussed. In Chapter 5, we delve into details about the implementation and experimental setup. Experimental results, detailed quantitative and qualitative analysis are performed by comparing them with existing techniques are reported in Chapter 6. In Chapter 7, we conclude and discuss the scope of this thesis.

CHAPTER 2

BACKGROUND STUDY

This chapter discusses the basic definitions and technical background of feature extraction and object detection techniques.

2.1 3D Model

A 3D model is a three-dimensional object's mathematical representation. Until it is represented, a model is not technically a graphic model. A model can be represented visually through a method called 3D rendering as a two-dimensional picture [3].

2.2 Rendering

The process of converting 3D wireframe models to 2D images automatically on a computer is called 3D rendering [3]. Advantages of 3D modeling over exclusively 2D methods include flexibility, ease of rendering, accurate photorealism, spatial reality, etc. Graphical Models are dependent on the synergy between computer graphics, computer vision and image processing. The traditional approach for generating virtual views of an object or a scene is to render directly from an appropriately constructed 3D model.

2.3 Virtual 3D city model

The Virtual 3D city model is a realistic and accurate representation of the environment in three dimensions. It is a geographically textured model of the surroundings where the vehicle navigates as shown in Figure 3. Such virtual 3D city models are produced using aerial/satellite imagery (photogrammetry), airborne laser scanner data (LIDAR), GIS and 3D computer graphic. There are different terms utilized for 3D city models in writing, for example, Virtual City', 'Cybertown', 'Cybercity', 'Digital City', '3D Urban Model'. The

need for 3D city models is growing and expanding rapidly in various fields, including urban planning and design, architecture, environmental visualization and many more [4].

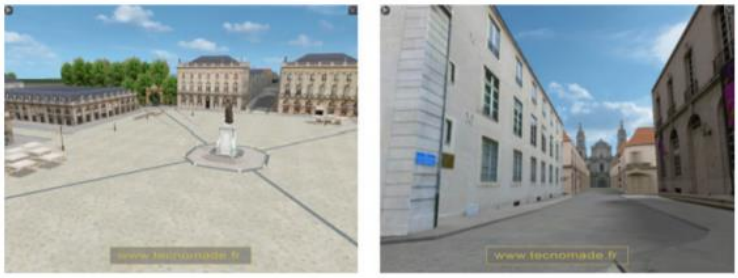


Figure 3: Views of the virtual 3D city model of Nancy [4]

2.4 3D Geographical Information System

To manipulate the virtual 3D city model, we need to navigate effectively in the 3D model by specifying the position and the orientation of the observer in a chosen reference frame. A computer tool has been developed for this purpose, referred to as the Three - Dimensional Geographic Information System (3D-GIS). A 3D-GIS can conceptualize terrain elevation, location of buildings, buildings facade texture, ground vegetation, rivers, etc. The inputs of the 3D-GIS are the 3D model database and the desired calibration parameters of the virtual camera. The 3D model database is composed of XML files containing the tagged information of every place with geo-locations, height and area information. The calibration parameters are the intrinsic parameters which are defined here as the horizontal field of view (FOV) on the one hand and the extrinsic parameters on the other. The latter being the position and the orientation of the virtual camera with respect to the frame that is attached to the 3D model [4]. The design of 3D-GIS is shown below in Figure 4.

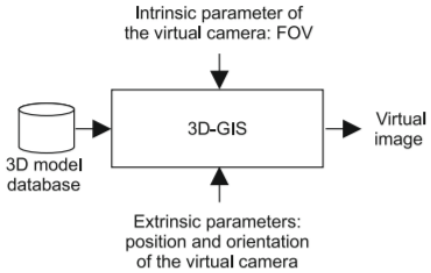


Figure 4: 3D-GIS [4]

2.5 Feature

A feature is described as “a piece of information which is relevant for solving the different computational tasks related to a specific application” [54]. Feature points are also referred to as keypoints/interest points/salient features. The idea of feature detection and description refers to the process of identifying points in an image (interest points) that can be used to describe the image’s contents such as edges, corners, ridges and blobs as shown below in Figure 5. Features are categorized into two standard categories: local features and global features. Local features are geometrical in shape, and global features are topological in shape [5]. It is mainly aiming towards object detection, analysis and tracking from a video stream to describe the semantics of its actions and behavior.

2.6 Feature Extraction

The term feature-detector (a.k.a extractor) traditionally refers to the algorithm or technique that detects feature-points in an image. Subsequently, the recognized characteristics are defined in logically distinct ways based on distinctive patterns that their adjacent pixels possess. This method is called the feature description as it describes each feature by assigning it a unique identity that allows for their efficient matching recognition. The terms detector and extractor are used interchangeably in this work [5].

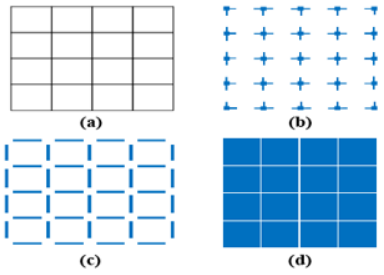


Figure 5: Illustrative image local features (a) input image (b) corners (c) edges (d) regions [5]

2.7 Feature Selection

Feature selection module is used for selecting a subset of relevant features from a large number of features extracted from the input data. The selected features are expected to

contain better discriminatory power to help distinguish among different classes with better accuracy. It also helps to reduce the dimensions of the feature space by selecting only the distinguishable features [56].

2.8 2D Feature Extraction techniques

Several techniques have been created for feature extraction and their operating patterns are quite distinct from each other as described in Figure 6. Each method's performance is optimal for a particular implementation.

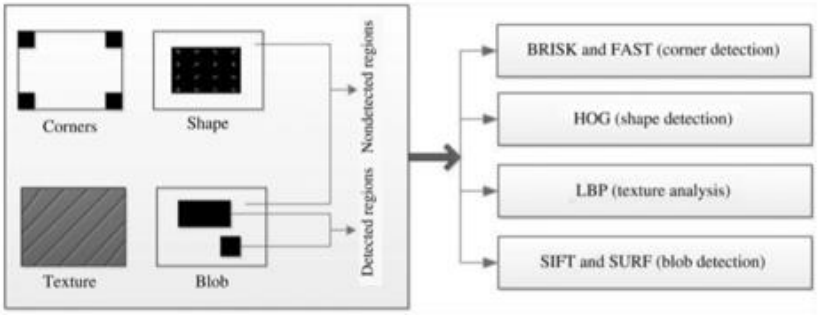


Figure 6: Common 2D Feature Extraction Methods [6]

Different types of 2D traditional Feature Extraction techniques are:

Local Binary Patterns (LBP):

It is an optimal feature extraction technique for texture analysis. It divides the image window into cells of 16 × 16 pixels, and every pixel in the cell is compared with eight of its neighbors, in which the center pixel has a higher value than other pixels. After the cell formation, a histogram is computed and normalized to make a feature vector. This feature vector can be processed by machine learning algorithms or SVM for classification. The improvements in LBP have enhanced the efficiency of face recognition applications, such as over complete LBP, transition LBP, modified LBP, and RGB-LBP, which are enriched with adjacent blocks overlapping, comparison of neighbor pixels, intensity values comparison of neighbor pixels, and computation of LBP for RGB-independent color

channels, respectively. Another outstanding combination of HOG-LBP has proved that the performance of LBP can be increased by combining it with other feature extraction algorithms [6].

Color histogram:

A color histogram is the representation of the distribution of colors in a picture [9]. For computerized pictures, a color histogram speaks to the number of pixels that have colors in each of a settled list of color ranges that span the image's color space, the set of all conceivable colors. Color histograms are adaptable builds that can be built from pictures in different color spaces, whether RGB or any other color space of any measurement. The main disadvantage of histograms for classification is that the representation is dependent on the object color, ignoring its shape and texture. Color histograms can be indistinguishable for two pictures with distinctive protest substance which happens to share color data. Then again, without spatial or shape data, comparative objects of diverse color may be undefined based exclusively on color histogram comparisons [7].

Canny Edge Detection:

A well-known, general and robust approach for edge detection in digital images was introduced by Canny. First, the input image is smoothed using a Gaussian filter. Subsequently, the values of the first derivatives in the horizontal and vertical direction are obtained by applying the Sobel operator to the smoothed input image. Using these values, the gradient magnitude and the edge direction can be calculated. The resulting edges are thinned using Non-Maximum Suppression (NMS). Subsequently, the remaining edge pixels are classified using a high and a low threshold in the so-called hysteresis. Edges above the high threshold are kept, edges below the low threshold are discarded. Edges between the low and the high threshold are only kept if there is an edge pixel within the respective 8-connected neighborhood [10]. This leads to a binary edge image as displayed in Figure 7.

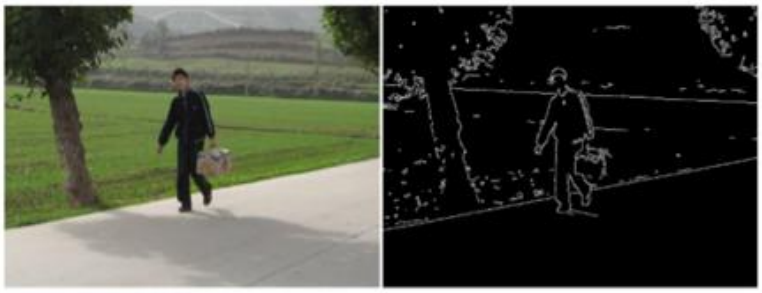


Figure 7: Left: Input image and Right: Canny edge image [10]

Structured Edge Detection (SED):

A more sophisticated, yet still real-time edge detection framework incorporating learning and the use of information of the objects of interest has been proposed by Dollár and Zitnick [71]. Therefore, in contrast to Canny edge detection, this approach requires a training procedure using an annotated training corpus. Here, a Random Forest (RF) maps patches of the input image I to output edge image patches using pixel-lookups and pairwise-difference features of 13 (3 colors, 2 magnitudes, and 8 orientation) channels. While testing, densely sampled, overlapping image patches are fed into the trained detector. The edge patch outputs which refer to the same pixel are locally averaged. The resulting intensity value (which lies in the interval [0,1]) can be seen as a confidence measure for the current pixel belonging to an edge. Subsequently, an NMS can be applied to sharpen the edges and reduce diffusion [10].

Scale Invariant Feature Transform (SIFT):

The SIFT method is used for extracting distinctive invariant features from images which will be used to perform reliable matching between different images using a nearest-neighbor algorithm. The significant steps in computation of SIFT are: 1) scale-space extrema detection and keypoint localization based on Difference of Gaussian (DoG) function to identify potential interest points; 2) orientation assignment to each keypoint location based on local image gradient direction; 3) the keypoint descriptor measures the local image gradients at the designated scale in the region around each keypoint [8].

Speeded-Up Robust Feature (SURF):

SURF is a local feature detection and matching method. The use of an integral image and basic Hessian-matrix approximation has dramatically reduced the computational complexity. The SURF parts consist of 1) interest point detection based on the Hessian matrix that approximates second-order Gaussian derivative with box filters by using integral images; 2) orientation assignment determined by constructing a circular region around the detected interest point and the dominant orientation describes the orientation of interest point; 3) interest point descriptors are built by extracting square windows around the interest points and computing the Haar wavelet responses in horizontal and vertical directions [8].

Histogram of Oriented Gradients (HOG):

Scale gradients, spatial binning, orientation binning, and contrast normalization are the key steps for human detection using HOG. According to gamma normalization, different color spaces that were used, such as LAB (LAB stands for Luminance/Lightness and A and B are chromatic components) and RGB, and grayscale gamma normalization has reduced the performance. This performance was evaluated by the false positive per window with respect to the miss rate. Log compression was found to be weak compared to the square root of gamma compression. Gaussian smoothing is used for gradient computation and different scales are tested. In orientation binning, a weighted vote was calculated for each pixel and these votes were summed up to make cells (orientation bins). The shape of cells is of two types: rectangular and circular. These orientation bins were equally spaced as 0 degree to 180 degrees (unsigned) and 0 degree to 360 degrees (signed). Contrast normalization and grouping blocks were performed for normalization. Two types of block geometries have been introduced as rectangular and circular, which are known as R-HOG and C-HOG, respectively [6].

Corner Detection Techniques

The term corner in detection does not mean to detect the physical corner such as the corner of the table or chair, but these are points in images with high curvatures. Corner means a point in the image whose gradient direction changes rapidly. The techniques of this class select the portion of the image which possesses the distinct properties from their immediate surroundings. Then, computes the key points or features which remain locally invariant or constant. Using these features, the image can be detected in different scenarios: rotation, scaling, translation and occlusion. Corner detection techniques are used for image recognition, detection and analysis [55]. List of type of detectors and descriptors present in Corner detection techniques are described in Figure 8 present below.

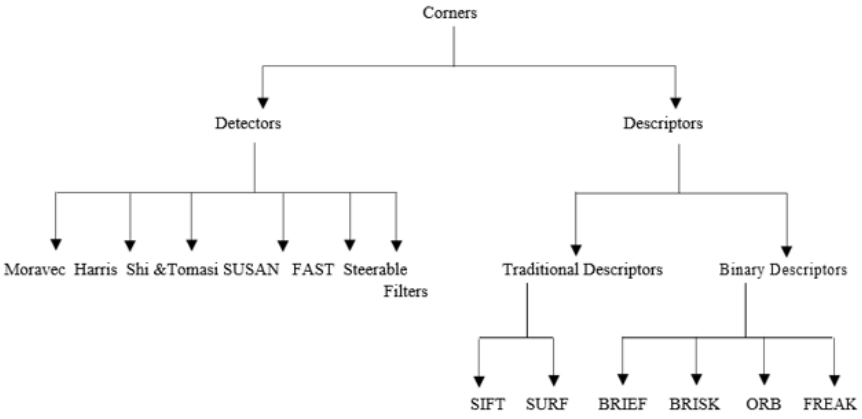


Figure 8: Classification of popular corner detectors and descriptors [54]

Forstner Corner Detector*:*

In 1986 Forstner presented a rotation-invariant corner identifier in light of the ratio between the determinant and the trace of μ. The accuracy of sub-pixel is used to find the location of a corner that is stable to a certain set of photometric and geometric transformations [54].

Harris Corner Detection:

Harris corner detection algorithm detects corners by forming a local search window and shifting it pixel-by-pixel in each direction. The variance in the pixel intensity helps the

algorithm identify peaks of low and high brightness levels. The center point of the window detects the corner. The shifting process averages the variation in pixel intensity. When the window is shifted along a flat or smooth part of the image where there is no drastic change in the pixel intensities, no corners are detected. However, when there is no change in intensity levels along the edge direction, an edge region is identified. When there is a significant change in intensity level in every direction, a corner is recognized [27]. Harris was successful in identifying robust features in any given image. But on account that it was only detecting corners, his work suffered from a lack of connectivity of feature-points which represented an essential obstacle for obtaining major level descriptors such as surfaces and objects [7].

Shi and Tomasi (Min Eigen) Corner Detection:

Shi and Tomasi have proposed the modified version of the Harris corner detector. This algorithm works in the almost same way like Harris but with a little change. Harris uses corner selection criteria with the help of Response Function R, if the score of R greater than a certain value, then the point will be called as a corner, where the score function computed by using two Eigenvalues. Shi & Tomasi have used Eigenvalues to decide corners instead of using score function [55]. It works quite well where even the Harris corner detector fails. We consider a small window on the image then scan the whole image, looking for corners. Shifting this small window in any direction would result in a large change in appearance if that particular window happens to be located on a corner. Flat regions will have no change in any direction. If there is an edge, then there will be no major change along the edge direction [57].

2.9 3D Feature Extraction techniques

3D keypoint detection is a critical step of object recognition. Several 3D keypoint detectors have been inspired by 2D feature engineering. The method for feature extraction should be independent of data representation. The method also should be invariant under transforms as translation, rotation and scale of a 3D object.

Different types of 3D Feature Extraction techniques are:

HOG 3D:

HOG3D is based upon histograms of oriented spatiotemporal gradients computed for a space-time volume in the neighborhood of an interesting point. This volume is further subdivided into video blocks. The gradient of each block is computed at different spatial and temporal scales, using integral video representation. Then, regular polyhedrons are used to uniformly quantize the orientation of the computed 3D gradients. The final gradient vector for space-time volume is obtained by concatenating the gradient vectors of all sub-blocks [58].

3D SIFT:

3D SIFT (Scale Invariant Feature Transform) extends the popular 2D SIFT to videos. The authors use finite difference approximations to compute the magnitude and orientations of 3D gradients for space-time volume around the interest points. Orientations of 3D gradients are parameterized by two angles: θ giving the gradient direction in 2D and φ encoding the angle away from the 2D gradient. The gradient magnitude is quantified along uniform orientations by dividing θ and φ into equally sized bins using meridians and parallels [58].

Harris 3D:

The Harris 3D detector is a space-time extension of the popular 2D (spatial) corner detector known as the Harris detector. It is a spatiotemporal interest point detector. In order to find spatiotemporal interest points, a second-moment matrix μ is computed for each video input point (x,y,t), at different spatial (σ) and temporal (τ) scale values. It uses a separable Gaussian smoothing function and space-time gradients. The descriptors used with Harris 3D are HOG/HOF descriptors. Harris3D might prove to be an inadequate representation of video by giving only a sparse set of spatiotemporal interest points. To overcome this limitation, dense sampling was introduced. Dense sampling extracts points at regular positions in time and space for different spatial and temporal scales [58].

However, each 3D object can be represented by a set of multiple rendered views as shown in Figure 9 instead of the original 3D model, some existing 2D image processing (feature extraction) methods can be employed. These images are captured with a static camera or virtual camera array [59].

Figure 9: 3D shape model rendered with different virtual cameras [72]

Ideal features should typically have the following essential qualities:

(1) Distinctiveness: The intensity patterns underlying the detected features should be rich in variations that can be used for distinguishing features and matching them.

(2) Locality: Features should be local to reduce the chances of getting occluded as well as to allow a simple estimation of geometric and photometric deformations between two frames with different views.

(3) Quantity: The total number of detected features (i.e., features density) should be sufficiently (not excessively) large to reflect the frame's content in a compact form.

(4) Accuracy: Features detected should be located accurately concerning different scales, shapes and pixels locations in a frame.

(5) Efficiency: Features should be efficiently identified in a short time to make them suitable for real-time (i.e., time-critical) applications.

(6) Repeatability: Given two frames of the same object (or scene) with different viewing settings, a high percentage of the detected features from the overlapped visible part should be found in both frames. Repeatability is greatly affected by the following two qualities.

(7) Invariance: In scenarios where large deformation is expected (scale, rotation, etc.), the detector algorithm should model this deformation mathematically, as precisely as possible so as to minimize its effect on the extracted features.

(8) Robustness: In scenarios where a small deformation is expected (noise, blur, discretization effects, compression artifacts, etc.), it is often sufficient to make detection algorithms less sensitive to such deformations (i.e., no drastic decrease in the accuracy) [5].

2.10 Computer Vision tasks

Computer Vision is the interdisciplinary scientific field that can recognize and understand images and scenes. As depicted in Figure 10, the three most common tasks in computer vision are the classification of an image, object classification with localization and object detection.

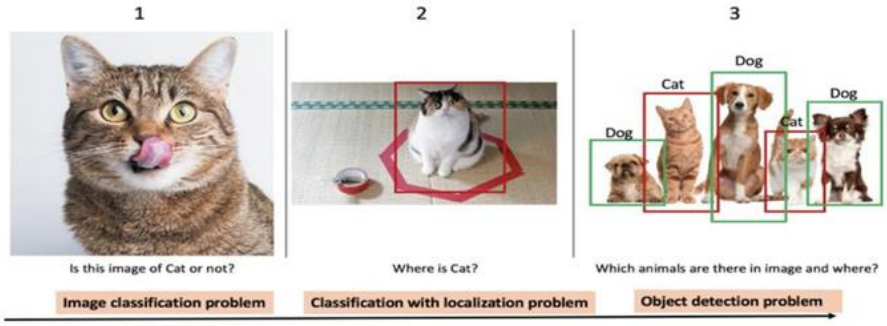


Figure 10: Three most common computer vision tasks [1]

Classification of an image:

Image Classification is the most common computer vision problem where an algorithm looks at a picture and classifies the object in it. Image classification has an extensive variety of applications, ranging from face detection on social networks to cancer detection in

medicine. Such problems are usually modeled using Convolutional Neural Nets (CNNs). Figure 11 shows the high-level steps involved in a typical image classification task. The input image is sent through multiple convolutional, pooling, non-linear layers and the output of the final layer of the CNN is passed into a softmax layer which converts the numbers between 0 and 1, giving the probability of the image being of a particular class.

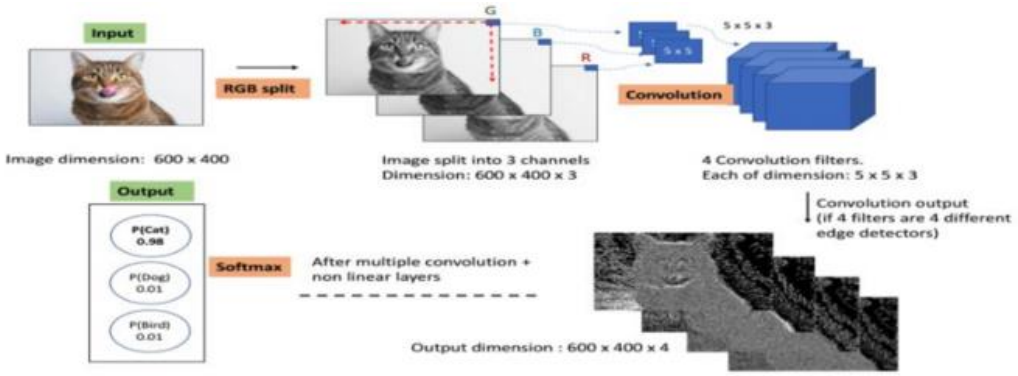


Figure 11: Steps for image classification using CNN [1]

Object classification and localization:

Localization of object's algorithms not only spots an object's class but also it draws a bounding box around an object's picture position as displayed in Figure 12. To get the bounding box location, four more numbers are added to the output layer. The final output contains class labels and four numbers to locate the bounding box.

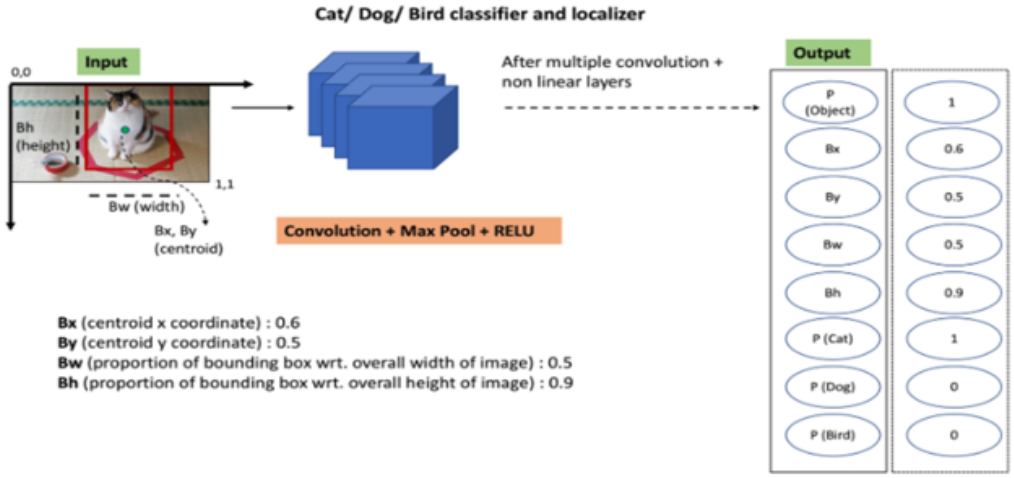


Figure 12: Input and Output for object localization problems [1]

Multiple objects detection and localization:

If multiple objects are present in the image and the task is to detect them all, then that would be a multiple object detection and localization problem. These kinds of issues need to leverage the concepts learned from image classification as well as from object localization. For the algorithm to detect all types of objects in an image, it needs to be capable of classifying and localizing all the objects in the picture as shown in Figure 13. This process is typically done using either a simple sliding window approach, wherein, the cropped window is passed through a ConvNet (Convolutional Neural Network) and have the ConvNet make the predictions. The sliding window is passed through the entire image. In the end, a set of cropped regions will remain, which will have some object, together with the class name and its bounding box. This sliding window approach is a fundamental object detection approach, and to tackle this problem, many advanced object detection algorithms have been devised [1].

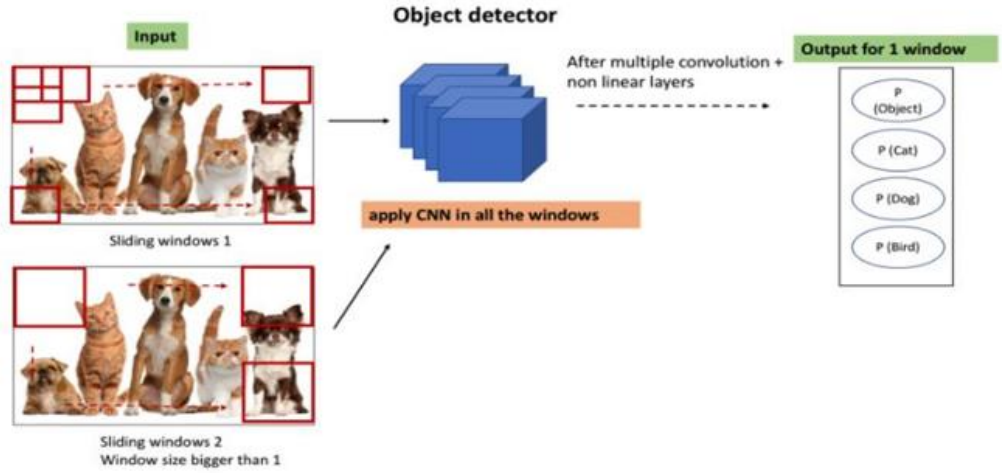


Figure 13: Multiple object detection and localization [1]

2.11 Object Detection Algorithms

Overview

Object detection algorithms work by finding out the specific object by matching the object's pixel values and equating them with the particular picture frame. There have been many object detection algorithms which have been proved to be feasible for engineering

uses. These algorithms have progressed from binary classified based approach to a learning-based approach [13].

2.11.1 Object Detection Traditional Methods

Viola-Jones Algorithm:

Viola-Jones algorithm was one of the breakthrough algorithms which was devised in 2001. It was mainly used for face detection but also was applicable for the general-purpose object detection techniques. It had four modules which included Haar Feature Selection, Integral Image Creation, Adaboost Training and Cascade Classifiers.

The algorithm searches for various features in a face like eyes, nose, mouth, etc. and computes face cascades and compares them with the Haar features to check for faces in an image. Due to this purpose, for the face to be detected, the images needed to be properly oriented with the face being frontal upright. This algorithm had very low false positives and very high detection rates. However, the recognition of faces was not quite developed as compared to detection rates, thus reducing practical implications [13].

Histograms of Oriented Gradients (HOG):

This algorithm interprets robust low-level features that are based on HOG. It is still grounded upon the approach of hardcoded features like the Viola-Jones method, but it is an alternative to exhaustive search. It initially converts to grayscale image and then finds the object by pixel-by-pixel in a particular frame. It matches each pixel with its surrounding pixels with reference to the intensity of darkness. By doing this, it can create a map of the gradients of the pixel intensity variation. These gradients can assist us to locate the nuances in an image. The HOG method computes the gradient orientation in localized portions of the image to identify multiple objects in a particular image. To highlight the required parts of the image and eliminate other background noise, feature descriptor can be used as presented in Figure 14. An image (size width x height x channels) to a feature vector/array of length n by the feature descriptor converts.

The feature vector produced by the algorithm produces excellent results when fed to an image classification algorithm like Support Vector Machine (SVM) [13].

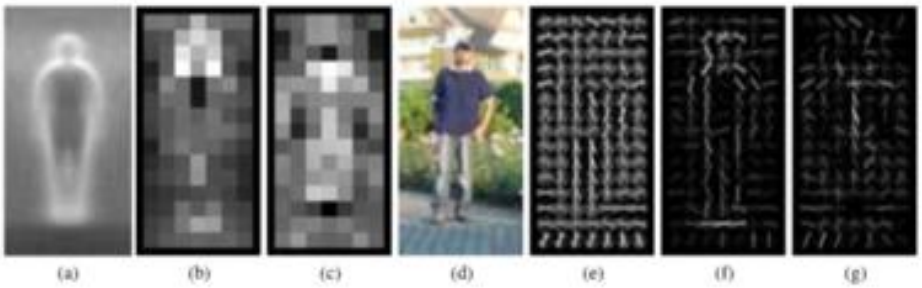


Figure 14: Illustration of HOG feature descriptors and SVM weights used for classification [74]

2.11.2 Deep learning methods

Deep learning is a component of a broader family of machine learning methods based on learning data representation [73]. Learning can be supervised, semi-supervised, or unsupervised. Deep learning models can attain state-of-the-art accuracy, sometimes surpassing performance at the human-level [15].

Deep learning architectures:

- Deep Neural Networks
- Deep Belief Networks
- Recurrent Neural Networks
- Convolutional Neural Networks

CNN

The basic building blocks of ConvNets (or CNN) are the convolutional layers, max-pooling or average pooling layers, and fully-connected layers. CNNs are connected as an arrangement of interconnected layers. The layers are made up of repeated pieces of convolutional, Rectified Linear Units (ReLU) and pooling layers. With a set of channels, the convolutional layers convolve their input. The channels are naturally found within the course of network training. The ReLU layer includes nonlinearity to the network, which

enables the network to memorize nonlinear combinations of the initial inputs, which is called feature extraction. These learned features, also known as activations, from one layer, become the inputs for the next layer. The pooling layers down-sample their inputs and help consolidate local image features. Finally, the learned features become the inputs to the classifier or the regression function at the end of the network. For image classification problems, the last layer is a classifier, and for object localization problems, the last layer is a combination of both [1]. The basic CNN architecture is represented in Figure 15. However, it is difficult to locate items precisely by directly mixing CNN with a sliding window approach. To address these issues, region-based CNN, that is, R-CNN, SPPnet and Fast-R-CNN have been proposed to improve object detection performance [17].

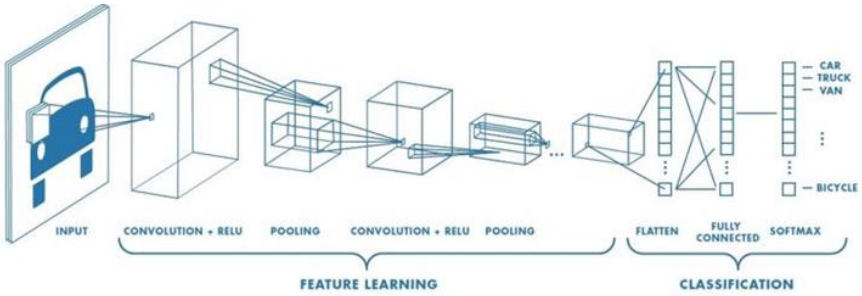


Figure 15: CNN Architecture [1]

R-CNN

R-CNN stands for Region-Based Convolution Neural Network and is a method that depends on the external region proposal system. R-CNN has proved to show better performance than other ensemble methods and feature types. R-CNN takes an input image and extracts region proposals and computes rich features using large CNNs and then classifies the image [13]. The basic R-CNN Architecture is shown in Figure 16. Although R-CNN was the new state-of-the-art system for general object detection, it is tough to identify small objects such as far-away cars and human faces, since the low resolution and lack of contexts in each candidate box significantly decrease the classification accuracy in them. Moreover, the two different phases in the R-CNN pipeline cannot be jointly optimized, leaving the trouble for applying end-to-end training on R-CNN [16].

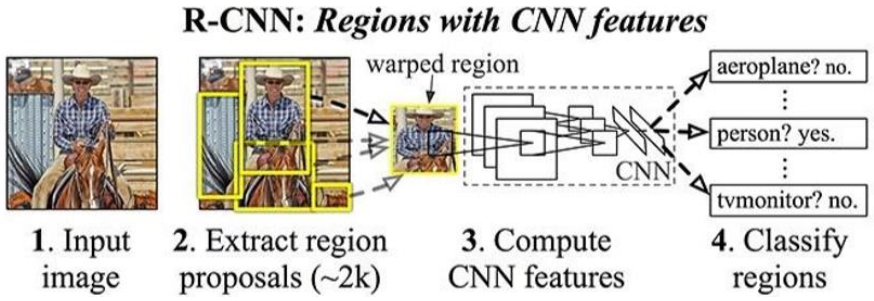


Figure 16: R-CNN Architecture [75]

FAST R-CNN

The main advantage of Fast R-CNN over previous state-of-the-art techniques lies in multi-stage pipeline training. In terms of space and time, training is expensive in R-CNN. On testing, it was observed that R-CNN based object detection was slow. R-CNN work is slowed down because of the execution of a ConvNet forward pass for each object proposal without sharing computation makes. The input image and a set of object proposals are fed into Fast R-CNN. It first processes the image through various convolutional and pooling layers and produces a convolutional map and then a fixed-length feature vector is extracted from the feature map for each proposal Region of Interest (RoI). To output, the K object classes by bounding boxes, each of these feature vectors are fed into a succession of fully connected layers that [13]. The basic Fast R-CNN Architecture is shown in Figure 17. The limitation of this approach is the prolonged computation time in the region proposal generation step. Therefore, Fast R-CNN was further improved by Ren et al., 2015 and Faster R-CNN was developed, which attained state-of-the-date object detection accuracy with real-time detection speed [17].

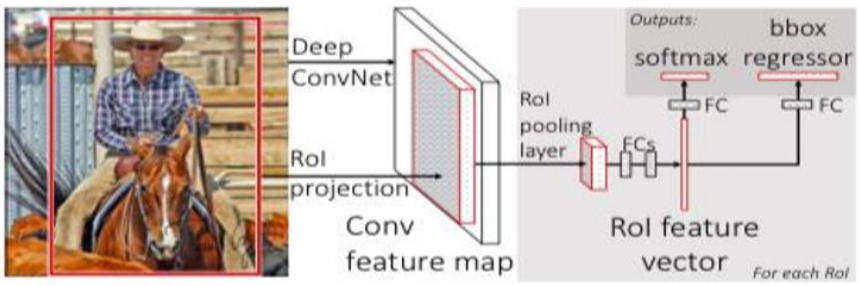


Figure 17: FAST R-CNN Architecture [20]

Recent Approaches:

Recent advances in self-driving cars have prompted researchers to build a variety of object detection algorithms. Most of these object detection algorithms are based on three meta-architectures: Single Shot multi-box Detector (SSD), Faster R-CNN (Regions with Convolutional Neural Networks) and Region-based Fully Convolutional Networks (R-FCN). Each of these architectures fundamentally differs in the way they build their object detection pipelines. A typical object detection pipeline can be mainly divided into three stages: informative region selection, feature extraction, and classification [1].

Single-Shot Detector (SSD)

The term single shot means the tasks of object localization and classification are handled in a single forward pass of the network. It is relevant to methods that require object proposals because it encapsulates all computation in a single network by eliminating proposal generation and subsequent pixel or feature resampling stages.

Based on a feed-forward convolutional network, a fixed-size group of bounding boxes and their corresponding scores for the target classification instances are shaped by this technique. Now by performing the non-maximum suppression step, the final detections are shaped. For high-quality image classification (with their last classification layer removed), the initial network layers are built on a standard architecture called as a base network (here it is VGG-16). The basic SSD Architecture is shown in Figure 18.

The SSD architecture builds on these base networks, by discarding the fully connected layers and replacing them with a set of auxiliary convolutional layers. These additional Convolutional layers enable the algorithm to progressively decrease the size of the input to each subsequent layer and extract features at multiple scales [1].

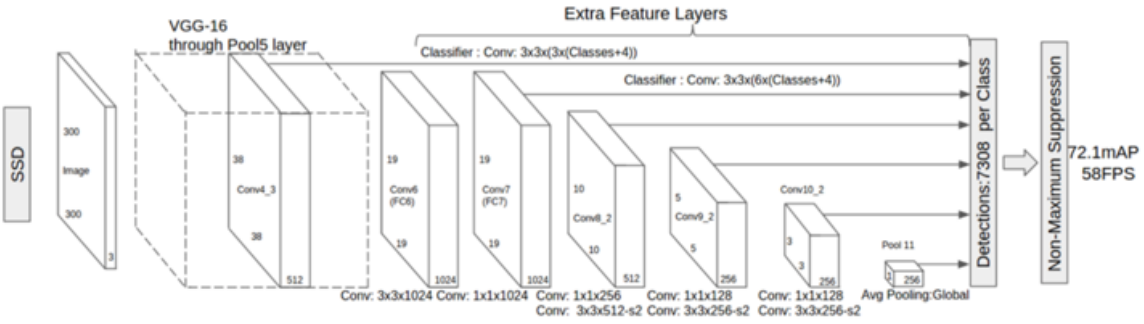


Figure 18: SSD Architecture [1]

Faster R-CNN

It consists of two networks one for region proposal network (RPN) (as shown in Figure 20) for generating region proposals and another for the network for detecting the object using these proposals. The main difference between Fast R-CNN is that it uses a selective search to create region proposals. As RPN shares the most computation with object detection, the time cost of making region proposals is much smaller in RPN than selective search. The region proposal network produces a cluster of boxes that are inspected by a classifier or a regressor to check for the occurrence of objects. After RPN, we get different sizes of proposed regions. Different sized regions mean different sized CNN feature maps. It's challenging to make an efficient structure to work on features of diverse sizes. A region of Interest Pooling simplifies the problem by reducing the feature maps to the same size. A fixed number of roughly equal regions (say k) are produced when an input feature map is divided with ROI splitting, and then Max-Pooling is applied to it. Therefore, the output of ROI Pooling is always k regardless of the size of the input. With the fixed ROI Pooling outputs as inputs, lots of choices are available for the architecture of the final regressor and classifier [13]. The basic Faster R-CNN Architecture is shown in Figure 19.

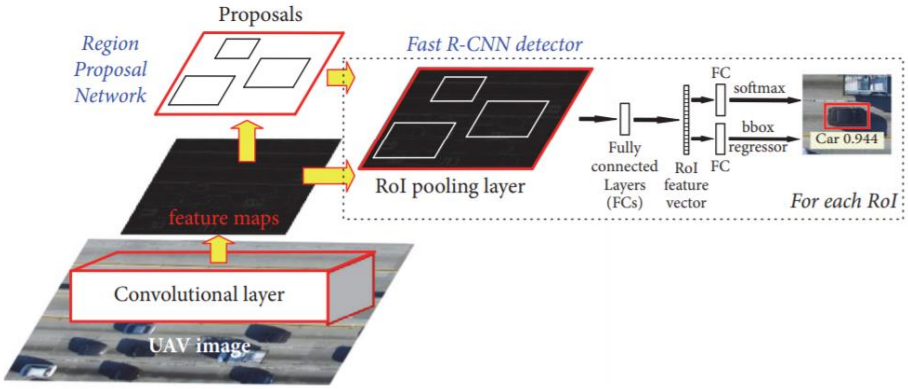


Figure 19: Faster RCNN Architecture [17]

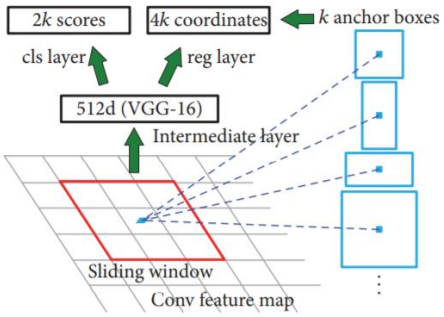


Figure 20: Region Proposal Network (RPN) [17]

Region-Based Fully Convolutional (R-FCN)

Region-based Fully convolutional networks introduced by Dai, J., et al. [75] provides an accurate and efficient object detection. R-FCN closely resembles the architecture of Faster R-CNN but instead of cropping features from the same layer where region proposals (RoI) are predicted, crops are taken from the last layer of features prior to prediction. Dai et al. argue this approach of pushing the cropping to the last layer greatly minimizes the amount of per-region computation that must be performed. This paper states that the R-FCN model (using Resnet 101) could achieve comparable accuracy to Faster R-CNN often at faster running times. The basic R-FCN Architecture is shown in Figure 21.

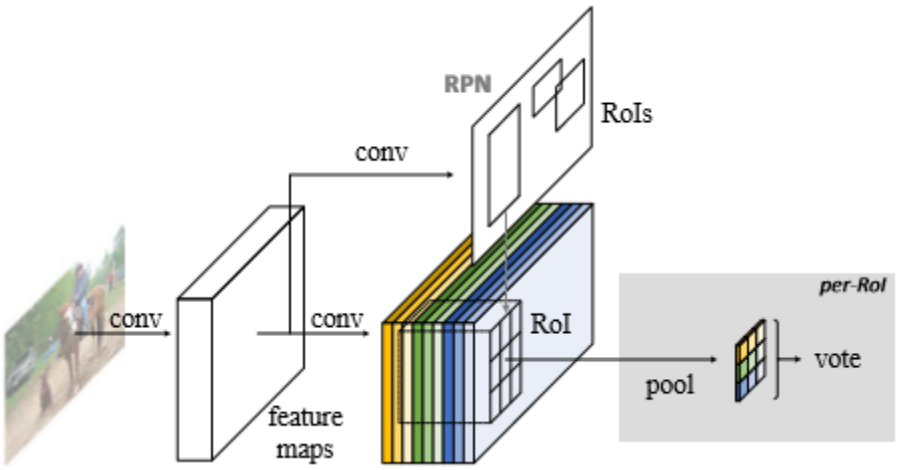


Figure 21: R-FCN Architecture [1]

In R-FCN architecture above, a given input object is divided into feature maps each detecting the corresponding region of the object. The feature maps are also known as position-sensitive score maps. Taking the example of the 3 X 3 ROI in Figure 21 above, we ask ourselves how likely each in the 3 X 3 matrix contains the corresponding part of the object and assign a score to it. This process of mapping score maps and ROIs to the vote array is called position-sensitive ROI-pool. The average of the resulting ROI pool gives the class score for a given object in the given ROI [1], as shown in Figure 22 below.

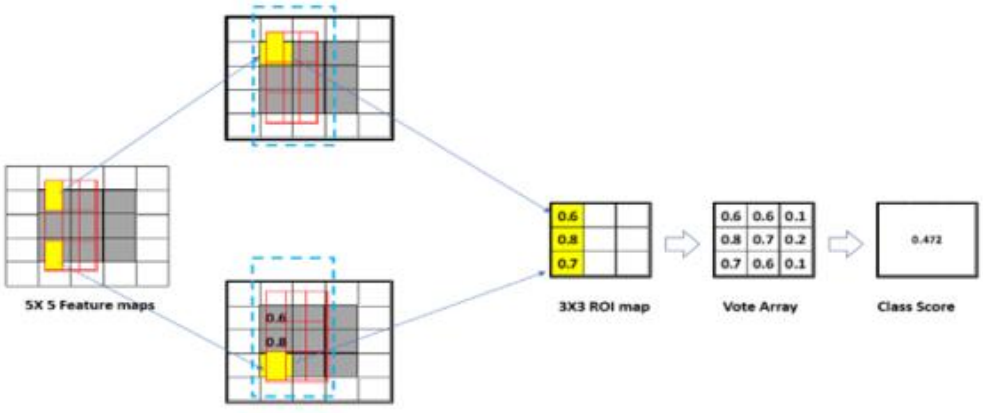


*Figure 22: Applying ROI onto the feature maps to output a 3*3 vote array*

Dilated CNN

The architecture of Dilated convolutions (à-trous convolutions/convolution with holes) is based on the fact that dilated convolutions support the exponential expansion of the receptive field without introducing additional parameters and sacrificing the resolution of data at the output layer and computational cost and filling the vacant positions with zeros to learn multi-scale features. In practice, no expanded filter is created instead, the filter features (weights) are matched to distant (not adjacent) features in the input matrix. The distance is determined via the dilation coefficient D. Generally, dilated convolutions have improved performance. With dilated convolutions in different dilation rates, receptive fields in various sizes can be obtained, those multi-scale features extracted are as displayed in Figure 23.

One of the critical components of our design is the dilated convolutional layer. A 2-D dilated convolution can be defined as follow:

$$y(m,n) = \sum_{i=1}^{M}\sum_{j=1}^{N} x(m + r \times i, n + r \times j)w(i,j)$$

y(m,n) is the output of dilated convolution from input x(m,n) and a filter w(i,j) with the length and the width of M and N respectively. The parameter r is the dilation rate. If r = 1, a dilated convolution turns into regular convolution.

In dilated convolution, a small-size kernel with k × k filter is enlarged to k + (k − 1)(r − 1) with dilated stride r. Thus it allows flexible aggregation of the multi-scale contextual information while keeping the resolution same [21]. They feature the scale of each convolutional kernel that is $(2l + 1)^2$ where l is the dilation rate of this kernel [22].

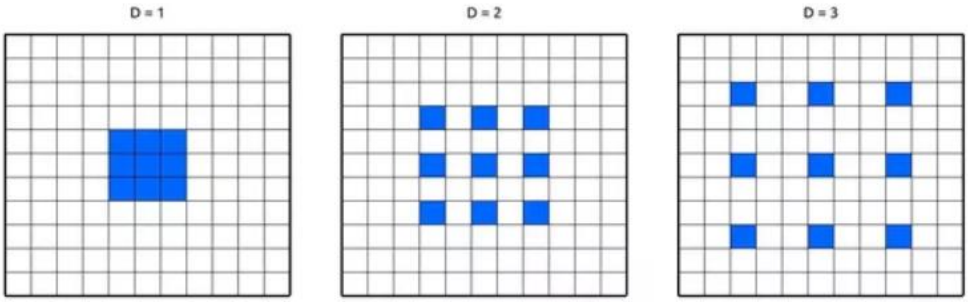


*Figure 23: An illustration showing the effect of increasing the dilation of 3*3 filter [31]*

In dilation1, there is a 1-pixel distance between each filtered pixel and its nearest neighbor. In dilation2, there is a 2-pixel distance: the filter is only applied to pixels that are in both odd-numbered rows and columns of each 5×5 region. In dilation3, the filter is applied to only pixels in every third row and column of each 7×7 region, and so on.

These dilations allow for a model to perceive higher-order abstractions without the need for dimensionality reduction. The main advantage of this type of mechanism is that the model is capable of capturing the frequency component from the input signal. Using this simple trick, the CNN can accommodate a larger receptive field also, thus letting the model accommodate longer-range dependencies. The dilated convolution technique opens up the possibility of making as many as required skips in the input data, so we have a better global view of the problem domain, in our case, one-dimensional vibration signal. It can alleviate the severe loss of spatial acuity due to multiple down samplings in a traditional deep network [24]. Another added advantage is if one wants to replace multiple dilated convolutional layers with a single convolutional layer with large size filter, it is safe. For instance, two 3×3 filters with 2-dilation can be replaced by one 9×9 filter with 1-dilation [25]. Dilated CNN can deliver 5×5 information with only nine weights instead of conventional CNN, which needs 25 weights [26]. However, a key drawback of dilated convolutions is exactly what they do not perform any subsampling, which is essential for reducing the complexity of deep layers. Thus, models that use dilation still rely on regular subsampling layers [23]. They are frequently used in model compression, audio generation, semantic image segmentation, generic image classification, sound wave synthesis, machine translation, signal processing, weakly-supervised object location, etc.

2.12 Heatmaps

Heatmap is a data matrix visualizing values in the cells by the use of a color gradient. This gives a good overview of the largest and smallest values in the matrix. Rows and/or columns of the matrix are often clustered so that users can interpret sets of rows or columns rather than individual ones.

In other words, a heatmap is a type of graphical representation of data that consists of a set of cells, in which each cell is painted for a specific color according to a specific value attributed to the cell. The term “heat” in this context is seen as a high concentration of geographical objects in a particular place. Heatmaps show the distribution of objects or phenomena across the entire surface. More generally, heatmaps can be viewed as the surfaces of densities. Such surface density well illustrates the location of the concentration of points or linear objects [37]. An example of a heatmap is shown below in Figure 24.

At a fundamental level, heatmaps are implemented as spatial matrices with cells colored after their values. Explicitly, they encode a continuous quantitative variable as a color in space through a color transfer function to a sequential color scheme [38].

Broadly speaking they fall into two classes: (i) image-based heat maps and (ii) data-matrix heat maps. Image-based heat maps display numerical information that is mapped over an image, an object or a geographic location. On the other hand, data-matrix heat maps display numerical data in a pseudo-colored tabular or matrix format. The data may be subsequently clustered using various measures of similarity or dissimilarity [39].

Figure 24: Heatmap generated on a sample image [28]

2.13 Annotations

From a technical point of view, annotations are usually seen as metadata, as they give additional information about an existing piece of data. We do not need a server to create local annotations. We can store annotation data in a local file system (local annotations - can be seen only by their owner) or it can store annotations remotely, on annotations servers accessed through the Web (remote annotations - can be seen by other people).

CHAPTER 3
LITERATURE SURVEY

This chapter discusses the relevant background of recent works in the Feature Extraction and Object Detection techniques

3.1 Static object feature extraction techniques:

Kaneva, B., et al. 2011 [30] proposed to use a photorealistic virtual world to gain complete and repeatable control of the environment to evaluate image features. They used two sets of images rendered from the Virtual City and from the Statue of Liberty to evaluate the performance of a selection of commonly-used feature descriptors, including SIFT, GLOH (Gradient Location and Orientation Histogram), DAISY, HOG, and SSIM (the Self-SIMilarity descriptor). They then used the virtual world to study the effects on descriptor performance of controlled changes in illumination and camera viewpoint resulting in the best performance of DAISY descriptor.

Cappelle, C., et al. 2012 [4] explored that in structured scenes, as indoor environments, line and edge are favored. But outdoor environments are often textured, so the usually used features are points, corners. In their work, the chosen features are the well-known Harris points. Once the Harris points are detected in the real image and in the virtual image, they have matched these two sets of points.

The Li, H., et al. 2018 [51] elucidates that when the number of images is too large, or the time of image feature extraction is long, it is not conducive to the implementation of the system. Aiming at this problem, this paper introduces and analyzes the principle and

shortcomings of Harris corner detection, and proposes an algorithm based on adaptive threshold Harris feature point selection is proposed. Firstly, the Harris algorithm is optimized from the two aspects of adaptive threshold and prescreening feature points. Then, in order to further reduce the pseudo-corner and prepare the image matching, the Forstner operator is used to determine the best feature point.

In [29], the algorithms compared by author are Moravec, Susan, Harris, FAST, Eigen and Forstner. The kinds of noise used are Gaussian, Poisson, salt & pepper and speckle. Results of testing the accuracy of each corner detector on the image noise mentioned that; a) not all corner detectors are able to find the corner points appropriately, b) all corner detectors do not show the location of the corner of the point accurately, c) all the corner detectors are very sensitive to all types of noise. The test results in this study show that the entire corner detector is very sensitive to noise, in other words, the degree of accuracy of detection results every corner detector will be strongly influenced by the noise and the type of noise

The commonly used key point descriptors like SIFT or SURF fail to obtain feature points [M. Yamaguchi et al., 2017]. Recently, new approaches for feature extraction based on deep learning methods [T. Faulhammer et al., 2016 and W. Kehl et al., 2016 and E. Simo-Serra et al., 2015] have demonstrated excellent performance.

3.2 Dynamic object feature extraction techniques:

In this Wang, Z., et al. 2017 [46] paper, the author elaborates that with orientation invariant feature embedding, local region features of different orientations are calculated based on 20 keypoint locations as mentioned in Figure 25 and are well aligned and combined. Firstly, vehicle images are propelled into the region proposal module, which produces the response maps of 20 vehicle key points. The key point regressor takes the input image and outputs one response map for each of the 20 key points. Instead of directly predicting boundary points or corner points, these key points are chosen as some discriminative

locations or some main vehicle components, e.g. the wheels, the lamps, the logos, the rear-view mirrors, the license plates.

1	left-front wheel	11	left rear-view mirror
2	left-back wheel	12	right rear-view mirror
3	right-front wheel	13	right-front corner of vehicle top
4	right-back wheel	14	left-front corner of vehicle top
5	right fog lamp	15	left-back corner of vehicle top
6	left fog lamp	16	right-back corner of vehicle top
7	right headlight	17	left rear lamp
8	left headlight	18	right rear lamp
9	front auto logo	19	rear auto logo
10	front license plate	20	rear license plate

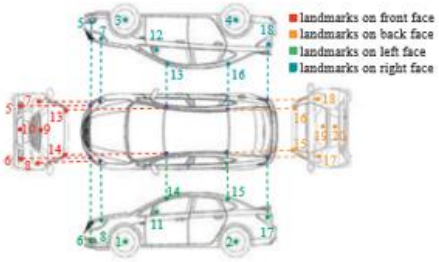


Figure 25: Definition and Illustration of 20 selected vehicle key points [46]

Chabot, F., et al. 2017 [47] goes on to explain that the Deep MANTA (Deep Many-Tasks), architecture consisting of the robust convolutional network provides vehicle part coordinates (even if these parts are hidden), part visibility and 3D template for each detection. They use a 3D vehicle dataset composed of 3D meshes with real dimensions. Several vertices are annotated for each 3D model. These 3D points correspond to vehicle parts (such as wheels, headlights, etc.) and define a 3D shape for each 3D model. The main idea of this approach is to recover the projection of these 3D points (2D shape) in the input image for each detected vehicle. Then, the best corresponding 3D model for each detection box is chosen as shown in Figure 26. For this purpose, they proposed a semiautomatic annotation process using 3D models to generate labels on real images for the Deep MANTA training. Labels from 3D models (geometry information, visibility, etc.) are automatically projected onto real images providing a large training dataset without labor-intensive annotation work.

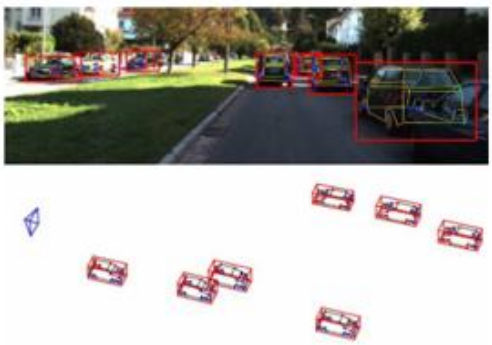


Figure 26: Corresponding 3D model for each detection box is shown [47]

Zhou, Y., et al. 2018 [34] states the possible reasons for the slow progress in Vehicle re-identification (re-ID) as the shortage of the special 3D structure of a vehicle and suitable research data. Previous works have mostly fixated on some specific views (e.g., front) but these methods are less effective in realistic situations, where vehicles usually appear in arbitrary views to cameras as explained in Figure 27. In this paper, the author focused on the uncertainty of vehicle viewpoint in re-ID, proposing two deep end-to-end architectures: The Spatially Concatenated ConvNet (SCCN) and Convolutional Neural Network (CNN)-LSTM Bi-Directional Loop (CLBL). Their models exploit the great advantages of the CNN and long short-term memory (LSTM) to learn transformations across different viewpoints of vehicles. Thus, a multi-view vehicle representation containing information about all viewpoints can be inferred from the only one input view and then used to calculate distance from learning. The output is presented in Figure 28. Experimental outcomes demonstrate that their models have achieved consistent improvements over the state-of-the-art vehicle re-ID approaches.

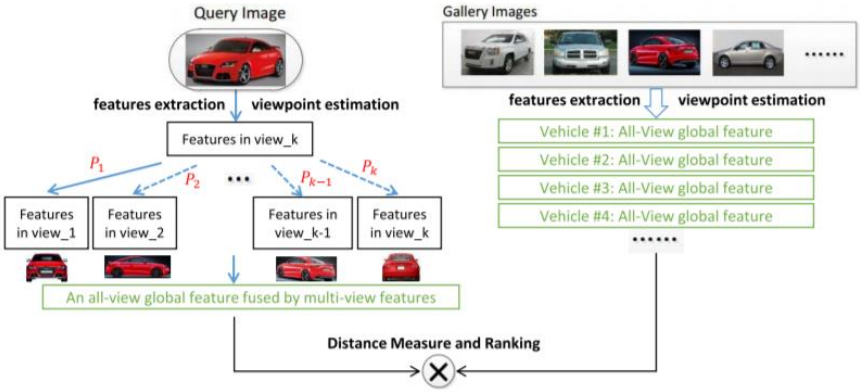


Figure 27: Motivation for multi-view vehicle re-ID [34]

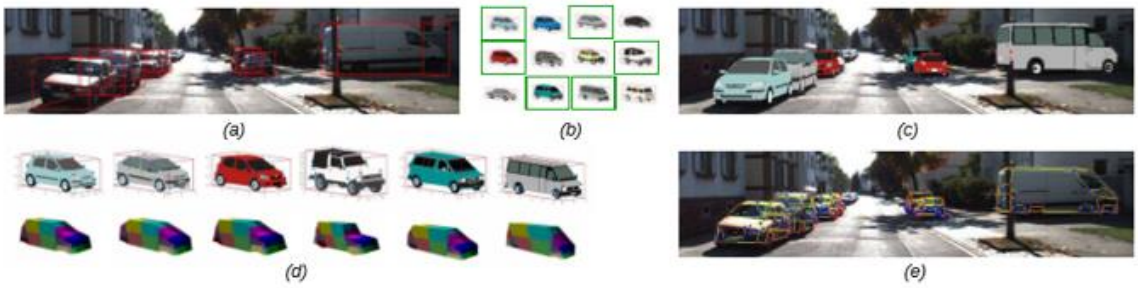


Figure 28: Semi-automatic annotation process [34]

To enable efficient labeling in 3D, Song, X., et al. 2019 [48] has built a pipeline by considering 2D-3D key point correspondences for a single instance and 3D relationship among multiple instances. To efficiently annotate complete 3D object properties, they have developed a context-aware 3D annotation pipeline. Nonetheless, KITTI only labels each car by a rectangular bounding box and lacks fine-grained semantic key point labels (e.g. window, headlight). In this paper, authors offer to the community the first large-scale and fully 3D shape labeled dataset. Besides, they contributed the first large-scale database suitable for 3D car instance understanding – ApolloCar3D, where each car is fitted with an industry-grade 3D (Computer-Aided Design) CAD model with the absolute model size and semantically labeled key points as presented in Figure 29.

Surface name	Keypoints label
Front surface	0, 1, 2, 3, 4, 5, 6, 8, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61
Left surface	7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
Rear surface	24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 62, 63, 64, 65
Right surface	36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50

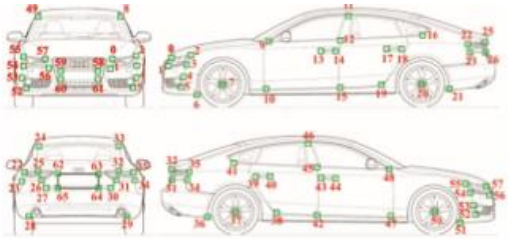


Figure 29: Defined 66 3D keypoints for car models [48]

This dataset is above 20× larger than PASCAL3D+ and KITTI, the current state-of-the-art. To enable efficient labeling in 3D, they build a pipeline by considering 2D-3D keypoint correspondences for a single instance and 3D relationship among multiple instances as pipelined in Figure 30. Complementing existing related datasets, we hope this new dataset could serve as a long-standing benchmark facilitating future research on 3D pose and shape recovery.

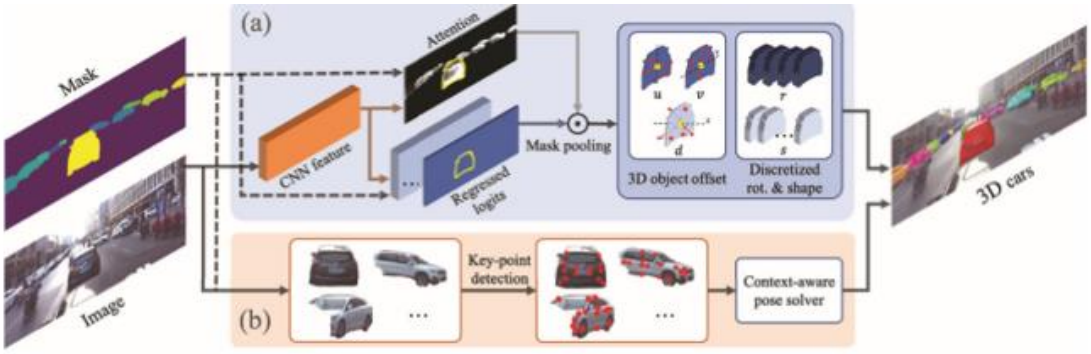


Figure 30: Training pipeline for 3D car understanding [48]

Warrington, A., et al. 2017 [53] states that the patch-based architecture uses strided max pool layers to reduce the spatial dimensions of the patch. However, applying this max pooling to a whole image reduces the dimensionality of the image, and hence the output will be of lower resolution than the input. A one-step method of circumventing this is to use unstrided max pool operators. However, this means that the effective field of view is dramatically reduced. Therefore, they have used dilated or atrous convolutions. Atrous convolutions 'inflate' the size of the mask by inserting fixed zeros at regular intervals. For instance, dilation of a three by three mask, with a dilation rate of one yields a five by five mask, with two rows and columns of zeros (resembling a noughts-and-crosses board). This allows us to capture the spatial extent similar to the original implementation while retaining a fully convolutional structure, but without exposing us to overfitting if we were to simply use larger, undilated convolutional kernels.

3.3 Static and Dynamic object feature extraction techniques:

In [59] authors had a set of depth images whose foreground has already been extracted. Laplacian operator is used for sharpening depth image then SIFT algorithm is adopted to get the key points. An algorithm called RANSAC (Random Sample Consensus) is employed as the filter before feature matching to improve the matching accuracy. The processing of feature extraction and feature matching is shown in Figure 31. However, the computation cost of feature extraction is still high.

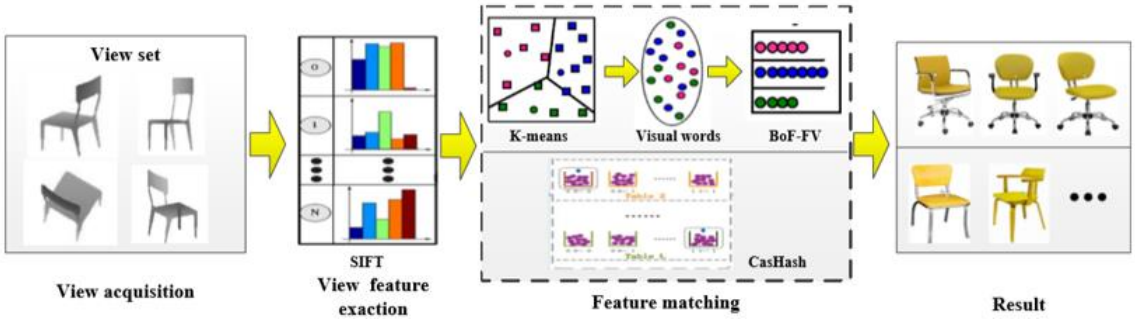


Figure 31: The process of the algorithm [59]

In [60] author proposed a Graphic Processing Unit (GPU) based algorithm that performs multi-view range image rendering and SIFT feature extraction of the Bag-of-Features (BF) SIFT algorithm on the GPU. This technique first calculates a set of multi-scale local visual features from a collection of depth images rendered from multiple view orientations about the 3D model. Thousands of visual features per model are combined into a feature vector for the model by using the bag-of-features approach. The algorithm performed very well, especially for models having articulation or global deformation. However, the method was computationally expensive because of the costs of rendering depth images, extracting local visual features and quantizing these features.

In [61] the author proposed an algorithm that samples each depth image of a 3D model more uniformly and densely. Such sampling would produce a "balanced" representation of local geometrical features in the bag of features, and consequently, in the histogram of visual words that describes the 3D model. They have chosen a dense and random sampling of the image, so the proposed method is called Bag-of-Features Dense SIFT (BF-DSIFT) algorithm. To extract and encode an increased number of local features efficiently, they have adopted a GPU implementation of the SIFT algorithm. The pipeline is explained in Figure 32.

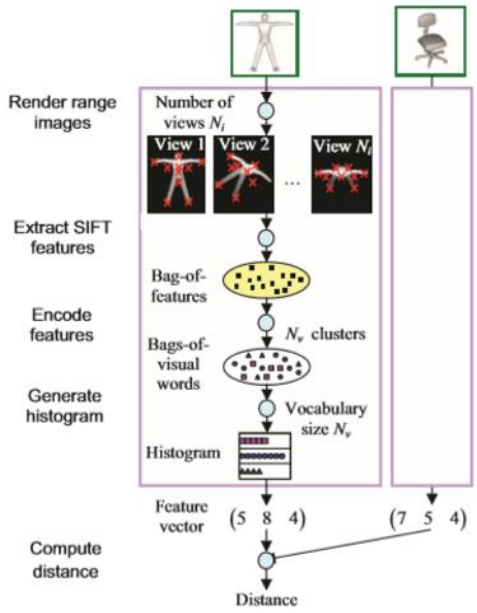


Figure 32: Local Visual features are integrated into a feature vector by using BoF approach [61]

In [62] authors adopted GPU to accelerate the procedure of feature extraction. Impressed by the superior performance of deep learning approaches in various visual tasks, they have proposed to use the activation of a Convolutional Neural Network (CNN). The CNN used here takes depth images as input, and the loss function is exerted on the classification error for projections. The network architecture consists of five successive convolutional layers and three fully connected layers. They have normalized each activation in its Euclidean norm to avoid scale changes. It only takes 56ms on average to extract the view features for a 3D model. The sequence flow of the feature extraction is described in Figure 33.

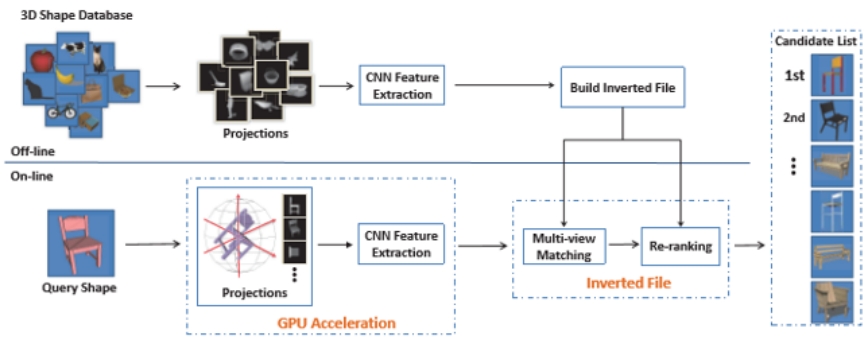


Figure 33: 3D shape search engine [62]

Gao, B. B., et al. 2015 [52] states that feature representation is among the most important topics in current state-of-the-art visual recognition tasks. Over the past decade, handcrafted features (e.g., SIFT and HOG) were very popular, and they were often encoded into a high dimensional vector by the Bag-of-Visual-Words (BOVW) framework. This representation is further improved by the Vector of the Locally Aggregated Descriptors and Fisher Vector methods, via adding higher-order statistics. However, such features are significantly outperformed by the recent deep features from CNNs which have exhibited significantly enhanced performance than those handcrafted features in visual recognition. In spite of the significant results achieved by deep features, there are many factors which can affect the performance of deep feature representations.

More recently, Aubry, M., et al. 2015 [19] proposed an approach which analyzes CNN feature responses corresponding to different scene factors by fully controlling them via rendering. To a trained CNN, the rendered snapshots are presented and responses for

different layers are studied concerning the input setting factors. The authors observed important differences across the CNN layers for various scene factors. They also demonstrated that their deep feature analysis based on computer-generated imagery is related to understanding the network representation of natural images.

Su, H., et al. 2015 [35] addresses that two issues hinder Object viewpoint estimation from 2D images progress: the scarcity of training data with viewpoint annotations and a lack of powerful features. They suggested a framework to address both issues by combining a scalable and over fit resistant render-based image synthesis pipeline from 3D models and CNNs to generate large-scale training data with fully annotated viewpoint estimate information. Critically, achieved this with negligible human effort. They presented that by cautiously designing the data synthesis process as shown in Figure 34, this method can significantly outperform existing methods on the task of viewpoint estimation on 12 object classes from PASCAL 3D+.

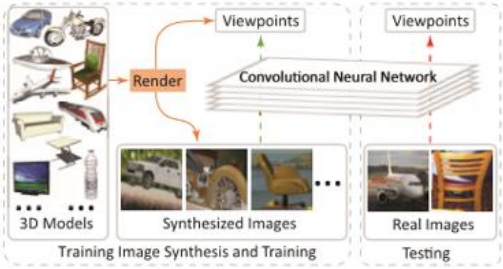


Figure 34: The learned CNN is applied to estimate the viewpoints of objects [35]

3.4 Static and Dynamic object detection techniques:

Region proposals are a cornerstone in the object detection methods. However, in YOLO (You Only Look Once), region proposition and classification are integrated into one single stage. YOLO's single-stage detection pipeline is extremely fast, making YOLO the first CNN based, general-purpose object detection model that achieved real-time speed when Compared with R-CNN and Faster R-CNN based methods [45]. The basic steps involved in YOLO are shown in Figure 35.

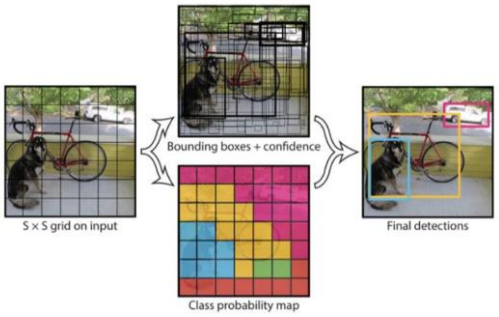


Figure 35: Illustration of YOLO [33]

Fully-convolutional networks (FCN) were popularized by Wu, B., et al. 2017 [45], who applied them to the semantic segmentation domain. FCN defines a broad class of CNNs, where the output of the final parameterized layer is a grid rather than a vector. This is useful in semantic segmentation, where each location in the grid corresponds to the predicted class of a pixel. FCN models have been applied in other areas as well. To address the image classification problem, CNN needs to output a 1-dimensional vector of class probabilities.

In Tang, S., et al. 2017 [32], the author mainly discusses different approaches on object detection methods. He says that detection with Deep Networks Region proposal methods plays an important role in object detection. Current top performing object detectors avoid exhaustive sliding window search across images and employ detection proposals to guide the search for objects. There are approaches mainly based on grouping, windows scoring, and CNN. Grouping proposal methods include methods such as Selective Search, Constrained Parametric Min-Cut, and Multiscale Combinatorial Grouping. Windows scoring methods are those that are based on objectness for sliding windows such as Objectness, EdgeBoxes, and Binarized Normed Gradients. CNN proposal methods consist of MultiBox, RPN, and HyperNet.

Selective Search that mainly employs color information merges superpixels to generate proposals. Edge Boxes that employs mostly the texture information starts from a coarse sliding window pattern builds on object boundary estimates and adds a subsequent refinement step to improve localization. Most recent faster R-CNN proposes RPN that mainly employs the supervised information to generate proposals [32].

3.5 Related Works:

Feature Extraction: Dynamic Objects (Cars)		
Paper	**Contribution**	**Limitations**
Suwajanakorn, S., et al. "Discovery of latent 3d keypoints via end-to-end geometric reasoning." 2018.	• Given a single 2D image of a known class, this network can predict a set of 3D key points that are consistent across viewing angles of the same object and across object instances. • These key points and their detectors are discovered and learned automatically without keypoint location supervision.	No information is provided regarding the estimation of orientation and the spatial transformation of the model.
Khan, S. D., et al. "Estimating Speeds and Directions of Pedestrians in Real-Time Videos: A solution to Road-Safety Problem." 2014	• This paper states that the driver must know when the pedestrian is going to change his/her walking direction to collision prone area. • Therefore, estimating the orientation of pedestrian on pedestrian crossing becomes very important to avoid such collisions.	Estimates orientation only for pedestrians. Hence the same concept can be applied to calculate for vehicles.

Feature Extraction: Static Objects (Buildings)		
Paper	**Contribution**	**Limitations**
Lal, K., et al. "Feature extraction for moving object detection in a non-stationary background." 2016	• The author states that using the Harris Corner Detection, the image turns out to be sharp with lesser noise and a significant number of points are detected. • They also concluded that the SURF algorithm was able to identify more points in the background comparing Harris and SIFT.	Harris Corner Detection and SURF Detector are able to process the information relatively faster than SIFT Detector, but Harris Corner Detection are seemingly more susceptible to noise and smooth pixel intensity level while SIFT Detector will detect the maximum amount of feature points, causing a longer processing time.
Xi, W., et al. "Comparisons of feature extraction algorithm based on unmanned aerial vehicle image." 2017	• The author presented the contrasts between point feature extraction operators like SIFT operator, Forstner operator, Harris operator by extracting feature points from the building images, grassland images, shrubbery images, and vegetable greenhouses images. • The extraction accuracy of the SIFT operator for the building is the lowest, while the extraction accuracy of the Forstner operator is the highest.	In the process of building feature points extraction, the Harris operator is slower. In the process of grassland feature points extraction process, Forstner operator is slower. In the process of bushes feature points extraction, the SIFT operator is slower. In the process of vegetable greenhouses feature points extraction, the Harris operator is slower.

Object Detection		
Paper	**Contribution**	**Limitations**
Ren, S., et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." 2015	• The author introduced a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. • An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. • RPNs are trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection.	The limitation is related to insufficient performance of Fast R-CNN block in Faster R-CNN. They have limitation for detecting relatively small objects in images. Most of the proposals are not always fit to target objects.
Jensen, M. B., et al. "Evaluating state-of-the-art object detector on challenging traffic light data." 2017	• The author applied the state-of-the-art, real-time object detection system You Only Look Once, (YOLO) on the public LISA Traffic Light dataset available through the VIVA-challenge, which contain a high number of annotated traffic lights, captured in varying light and weather conditions.	In this paper, YOLO is applied only on the daytime data from the freely available LISA (Laboratory for Intelligent and Safe Automobiles) Traffic Light Dataset. YOLO doesn't provide any information regarding the detected color.

CHAPTER 4
PROPOSED METHODOLOGY

This chapter discusses the proposed system and flowcharts for executing the feature extraction techniques and other modules. Furthermore, this section discusses the working of the overall system and connection of the proposed approach with all other modules.

4.1 Proposed Methodology Overview

Our idea is to find a method of directly using graphic models created from open-source datasets as the virtual representation of real-world objects. Our approach uses Machine Learning techniques to extract 3D feature points and to create annotations from graphic models for the identification of dynamic objects, such as cars, and for the verification and elimination of stationary objects, such as buildings.

In our method, a Dilated Convolutional Neural Network model is used to detect feature points of 3D car models. This helps to generate spatial transformation and estimate the orientation of the key points. This information is required for object identification.

As the pre-trained model is not available for trees and buildings, Faster RCNN trained model is constructed using open google images.

A faster RCNN approach is used to detect buildings in the input road scene and virtual scene. Annotations are generated for 3D virtual building models using corner feature extraction algorithms (FAST algorithm) followed by a feature selection technique for the object verification purpose.

A Priori Knowledge of 3D virtual buildings is used for the generation of heatmaps on real-time data for the elimination purpose because the cost and the computation time for the dynamic object recognition can be improved by the self-driving system.

4.1.1 Training Information

Cars:

- Dataset used: ShapeNet dataset
- Train set: 149 models
- Test set: 18 models
- Validation set: 18 models

Buildings:

- Dataset used: Google OpenImage dataset
- Train set: 2000 rendered images
- Test set: 400 rendered images

Trees:

- Dataset used: Google OpenImage dataset
- Train set: 1000 rendered images
- Test set: 400 rendered images

Training Time (in single GPU):

Cars: 20 days

Buildings and Trees: 3 days

4.2 CARS

Algorithm for Feature Extraction

Input: 185 3D models of cars (.obj)

Output: Annotation files containing Normalised 2D keypoints, Orientation estimation and spatial transformations of all rendered images (.txt files)

1) Perform Rendering to generate 240 images each of size 128*128 of each model
2) Create. tfRecords for each model.
3) Out of which, 149. tfRecords are used as the training set, 18 as the test set and remaining 18 as the validation set. Define a 4*4 global camera projection matrix
4) Send the output of step 4 and step 5, along with tuned parameters such as count of keypoints, epoch value, etc. to dilated CNN.
5) Now output generated by the convolutional dilated base network is sent to Orientation network. Orientation network constructs a system that infers the orientation of an object.
6) After this, with the help of orientation network, keypoint network is built, which predicts the 3D keypoints. The ground-truth orientation flag is used at the beginning of training. Then we linearly anneal in the prediction. Anneal refers to a number between [0, 1] where 1 means using the ground-truth orientation and 0 means using our estimate.
7) Now Rendered images are sent as test images to Keypoint Net to predict 3D keypoints.
8) Once 3D keypoints are predicted, they are normalized to 2D keypoints.
9) With the help of these normalized 2D keypoints, orientation and spatial transformation (using Euclidean distance) of the key points are generated.
   Orientation: $\tan\Theta = y/x$ where (x,y) is the normalized keypoint location.
10) Outputs are saved as annotation files(.txt) in the repository for feature matching and object identification.

Flowchart for Feature Extraction:

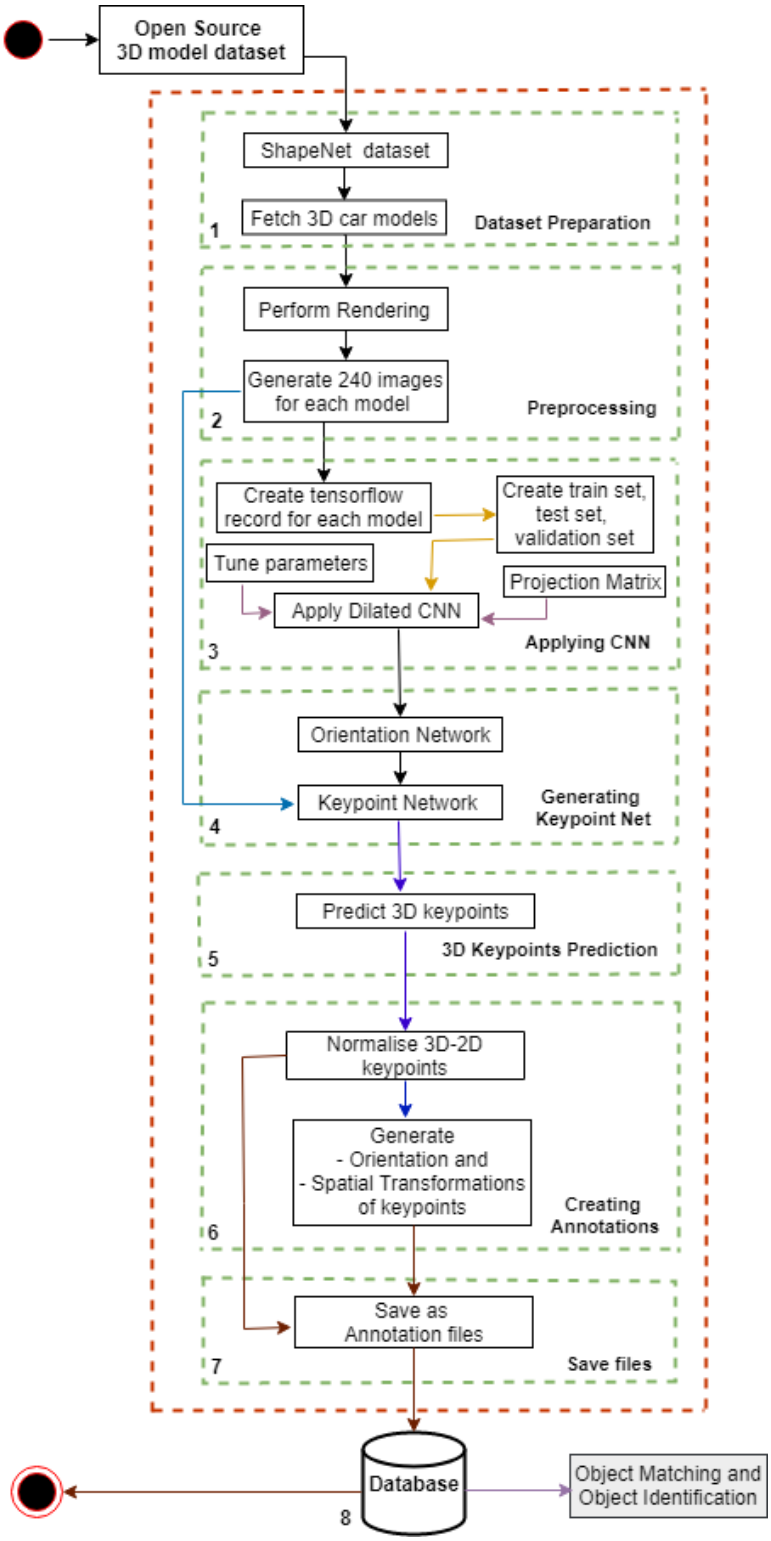


Figure 36: System Architecture for Feature Extraction of car models

Flowchart Explanation:

In the feature extraction module of cars, we first select 3D models of cars (.obj) from any open-source dataset say ShapeNet dataset. As a part of the experiment, I have selected 185 car models. For each model, we normalize the object so that the longest dimension lies in between [−1, 1]. After which rendering is done to generate 240 images each of size 128*128 for each model. Then we create .tf records for each model. Out of these, 149. tfRecords are used as the training set, 18 as the test set and remaining 18 as the validation set. Next, we define a 4*4 global camera projection matrix. The projection matrix, the generated .tf records and tuned parameters such as count of key points to be detected, epoch value, etc. to dilated CNN. Here we have selected the number of key points to be detected as 10, the total batch size of 256 epoch values as 200k using synchronous training with 32 replicas, etc.

Now output generated by the convolutional dilated base network is sent to Orientation network. Orientation network constructs a system that infers the orientation of an object. With the help of orientation network, keypoint network is built, which predicts the 3D keypoints. The ground-truth orientation flag is used at the beginning of training. Then we linearly anneal in the prediction. Anneal refers to a number between [0, 1] where 1 means using the ground-truth orientation and 0 means using our estimate. Now Rendered images are sent as test images to Keypoint Net to predict 3D keypoints.

Once 3D keypoints are predicted, they are normalized to 2D keypoints. With the help of these normalized 2D keypoints, orientation and spatial transformation (using Euclidean distance matrix) of the key points are generated. A spatial transformation is a mapping function that establishes a spatial correspondence between all points in an image and its warped counterpart. If A is a Euclidean distance matrix and the points x1, x2, x3…., xn are defined on m-dimensional space, then the elements of A are given by A = (aij) where aij = $dij^2 = ||xi-xj||^2_2$. Orientation: $\tan\Theta = y/x$ where (x,y) is the normalized keypoint location. Outputs are now saved as annotation files (.txt) in the repository for feature matching and object identification.

4.3 BUILDINGS

Algorithm for Feature Extraction

Input: Textured 3D model of Virtual city (.obj)

Output: 32 feature points for each rendered image (.txt file)

---

1) Perform rendering to fetch building virtual view scene.
2) Now, Calibrate these virtual views.
3) Perform Object detection technique like Faster RCNN to detect buildings.
4) Once identified, its corresponding bounding box coordinates are retrieved.
5) Based on those coordinates, cropping task is performed otherwise go to next frame.
6) Perform preprocessing on the cropped images.
7) A FAST algorithm is applied as part of the feature extraction (corner detection) technique.
8) Apply non-maximal suppression.
9) It generates a robust list of feature points.
10) Once these set of feature points generated, the center of each cropped image is calculated and Euclidean distance formula is applied with respect to each feature point for an image and calculated center.

    Euclidean formula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
    where the points (x1, y1) and (x2, y2) are in 2-dimensional space.

11) Now 32 key points who have longest distances from the center are selected as the optimal set of keypoints. These points act as selective features.
12) And now these points are saved as annotation as (.txt file) are saved in the repository for feature matching and object verification.

Feature Detection using the FAST algorithm

Input: Rendered image (.png/.jpg)

Output: feature points for each rendered image

1) Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be I_p.
2) Select an appropriate threshold value t.
3) Consider a circle of 16 pixels around the pixel under test. (This is a Bresenham circle of radius 3) as shown in figure 37.
4) Now the pixel p is a corner if there exists a set of n(=12) contiguous pixels in the circle (of 16 pixels) which are all brighter than $I_p + t$, or all darker than $I_p - t$.
5) A high-speed test was proposed to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If p is a corner, then at least three of these must all be brighter than $I_p + t$ or darker than $I_p - t$. If neither of these is the case, then p cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle.
6) Now we should apply Non-Maximum Suppression to avoid detecting multiple interest points in adjacent locations.
7) Compute a score function, V for all the detected feature points. V is the sum of the absolute difference between p and 16 surrounding pixels values.
8) Consider two adjacent keypoints and compute their V values.
9) Discard the one with lower V value
10) Display the robust set of keypoints

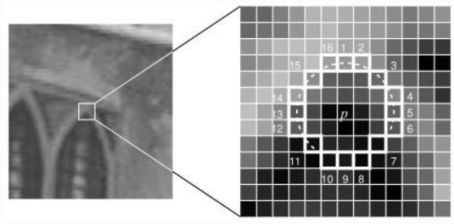


Figure 37: A circle of 16 pixels around the pixel under test [70]

Flowchart for Feature Extraction:

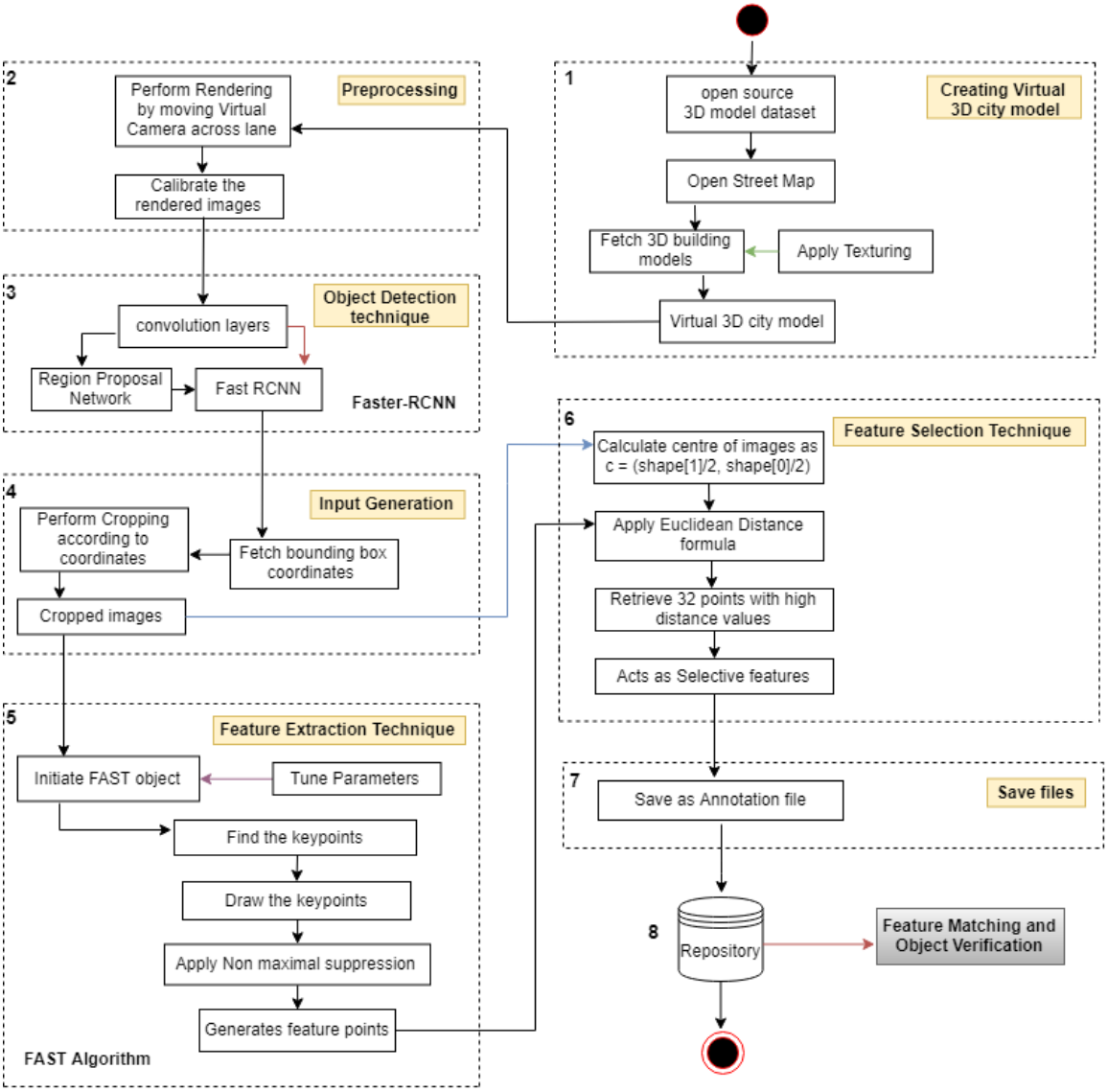


Figure 38: System Architecture for Feature Extraction of building models

Flowchart Explanation:

As a part of Feature Extraction of buildings, the input is the video of the textured 3D model of the virtual city. Perform rendering to fetch building virtual view scene. Then, we calibrate these virtual views. Now, perform Object detection technique like Faster RCNN to detect buildings. In faster RCNN the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using an RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

If the desired object is not present, then go to the next frame. Or else, once identified, its corresponding bounding box coordinates are retrieved. Based on those coordinates, cropping task is performed otherwise go to next frame. Next, perform preprocessing on the cropped images. After this, the FAST algorithm is applied as part of the feature extraction (corner detection) technique to find the key points. The advantage of the FAST corner detector is its computational competence. True to its name, it is actually faster than many other well-known feature extraction methods such as SIFT and Harris detectors. Moreover, the FAST corner detector is very suitable for real-time video processing application because of this high-speed performance. Next, apply Non-Maximal Suppression step which generates a robust list of feature points. Once these set of feature points generated, center of each cropped image is calculated and Euclidean distance formula is applied with respect to each feature point for an image and calculated center.

Euclidean formula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

where the points (x1, y1) and (x2, y2) are in 2-dimensional space

Now retrieve 32 key points which have the longest distances from the center are selected as the optimal set of keypoints. These points act as selective features. Finally, these points are kept as annotations (.txt file) and are saved in the repository for feature matching and object verification.

Buildings

Algorithm for heatmap generation

Input: Textured 3D model of Virtual city (.obj)

Output: Binary Heatmap of the rendered scene

---

1) Perform rendering to real-time scene
2) Now, Calibrate these views
3) Perform Object detection technique like Faster RCNN to detect buildings
4) Once identified, its corresponding bounding box coordinates are retrieved
5) Based on those coordinates, cropping task is performed
6) Perform preprocessing on the cropped images.
7) Remove noise by eliminating background details like the sky, etc
8) Fetch its corresponding structural information like height and width from virtual data and perform scaling accordingly
9) Execute applyColorMap() to its grayscaled image
10) Now generate heatmap by tuning parameters
11) Perform heatmap resizing according to the virtual structural information
12) If heatmap_pixel_value == (128,0,0)

    Then replace heatmap_matrix_value == 0

    Else replace heatmap_matrix_value == 1
13) Now binary heatmap is generated
14) Finally, save binary heatmap in the repository for Object Elimination

Flowchart for heatmap generation:

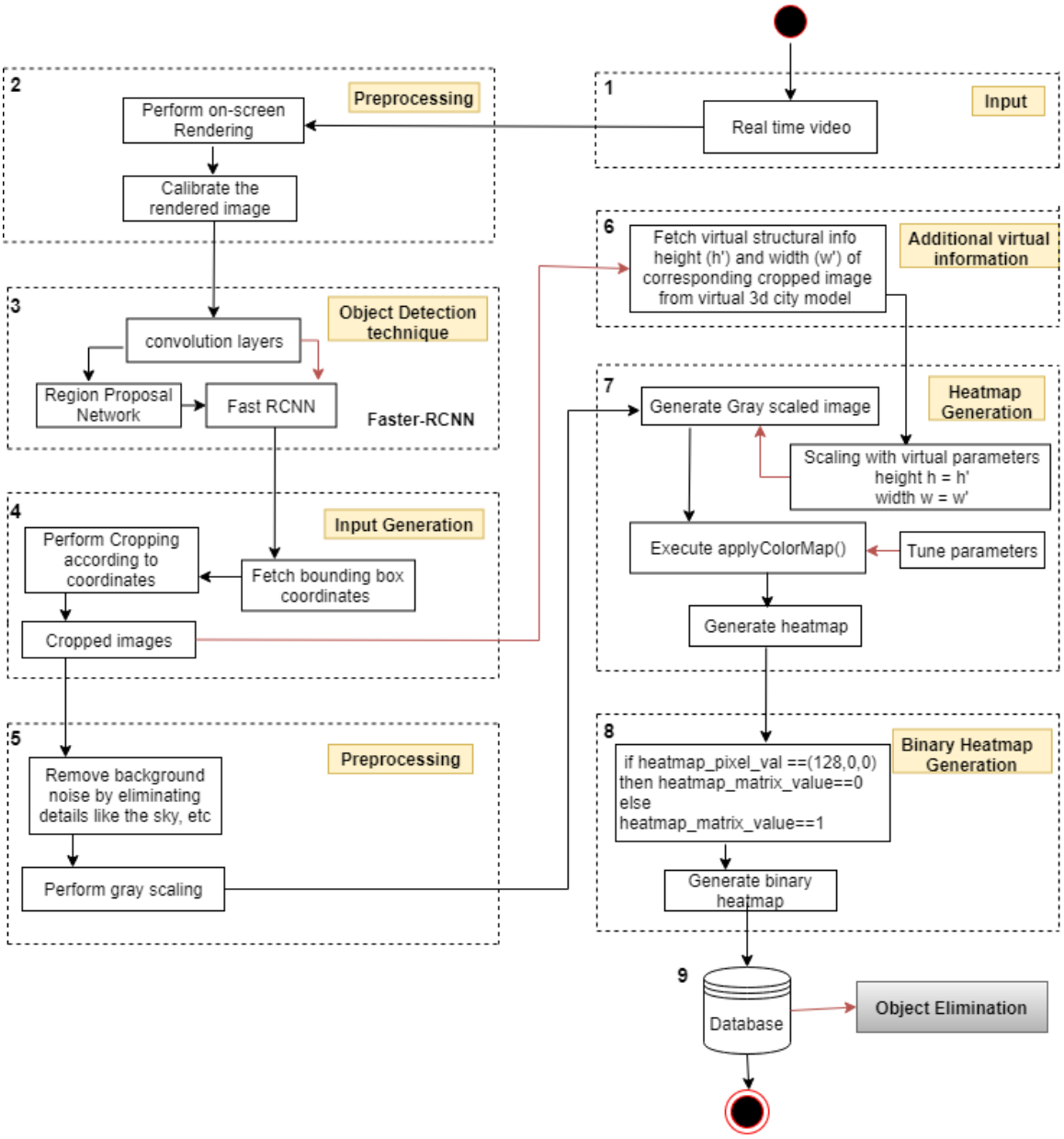


Figure 39: System Architecture for Heatmap generation for buildings

Flowchart Explanation:

In the heatmap generation module of buildings, the input is the real-time we first perform rendering to the real-time scene. Then we calibrate these views. Next, we perform Object detection technique like Faster RCNN to detect buildings. In faster RCNN the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using an RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. Once identified, its corresponding bounding box coordinates are retrieved.

Based on those coordinates, cropping task is performed. Now, perform preprocessing on the cropped images. After that, remove noise by eliminating background details like the sky, etc. Then, fetch its corresponding structural information like height and width from virtual data and perform scaling accordingly. Followed by this step, we now execute applyColorMap() to its grayscaled image. A colormap is a mapping from 0-255 values to 256 colors. In OpenCV, we can create an 8-bit color image of size 256 x 1 to store the 256 color values.

Now generate heatmap by tuning parameters. Next, perform heatmap resizing according to the virtual building structural information. The height of the building is present in the OSM file but the width needs to be calculated. The width is calculated using its realtime footprint coordinates. The height and width of each building are stored along with its place_id, osm_id, latitude, longitude and address.

If heatmap_pixel_value is (128,0,0) then replace heatmap_matrix_value as 0 or else with 1. Now generated heatmap is called as binary heatmap as it contains only binary digits. Finally, save binary heatmap in the repository for Object Elimination.

4.4 TREES

Algorithm for Trees 3D reconstruction

Input: Textures of trees from real-time data

Output: Reconstructed 3D tree

---

1) Perform rendering to real-time scene
2) Now, Calibrate these views
3) Perform Object detection technique like Faster RCNN to detect trees
4) Once identified, its corresponding bounding box coordinates are retrieved
5) Based on those coordinates, cropping task is performed
6) Perform preprocessing on the cropped images
7) Remove noise by eliminating background details like the sky, etc
8) Now calculate the height and width of each tree texture
9) Open blender and create a plane
10) Now resize the plane using the calculated structured information
11) Add material and apply the texture to the plane
12) Tune parameters like alpha etc
13) Perform UV mapping if necessary
14) The output now generated is a 3d tree
15) Thus this can be used to update virtual city to add more realism to it

Flowchart for 3D Tree reconstruction:

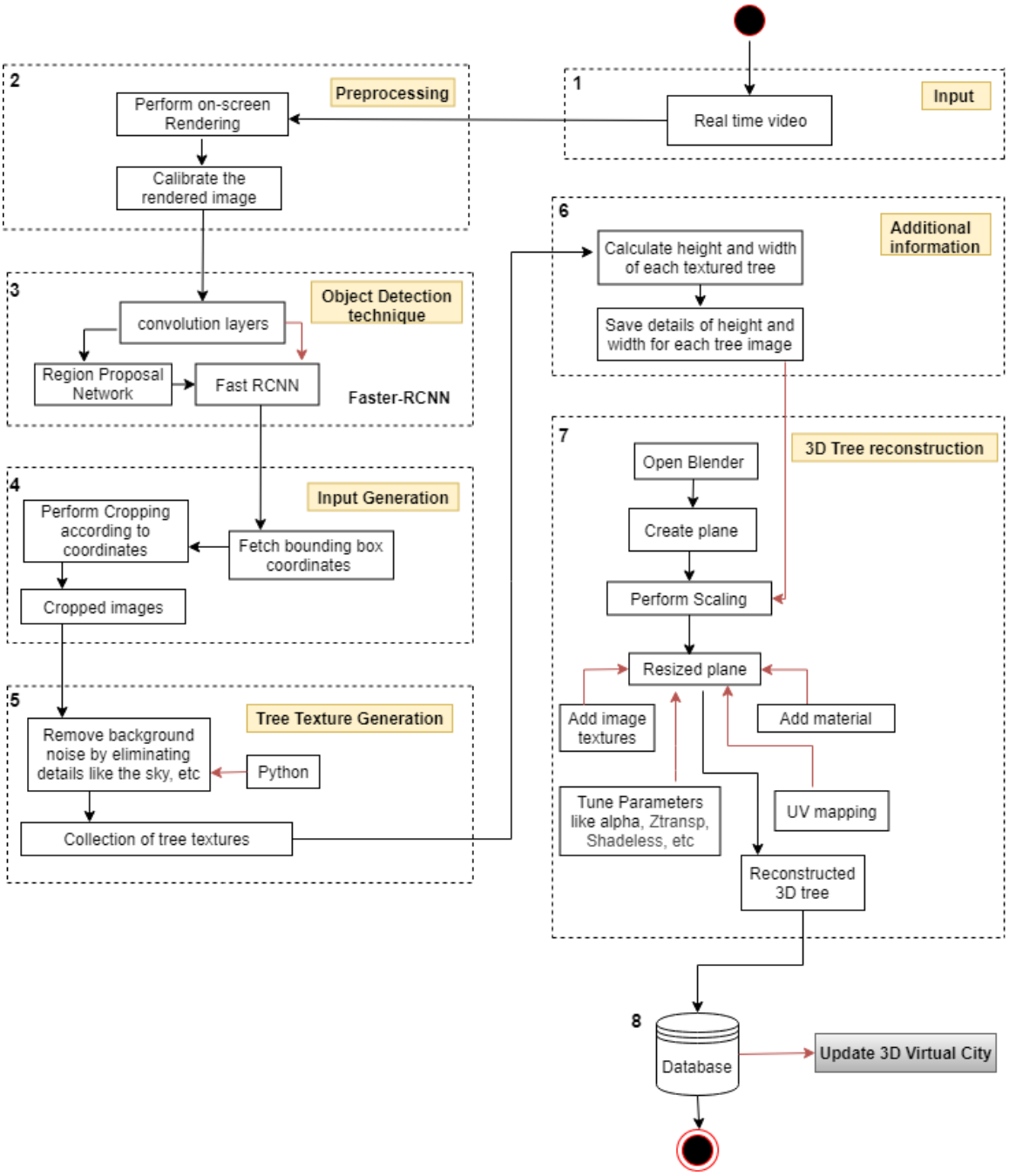


Figure 40: System Architecture for 3D Tree reconstruction

Flowchart Explanation:

In 3D Tree reconstruction module, the input is real-time video. We first perform rendering to fetch and calibrate these views. Next, we perform Object detection technique like Faster RCNN to detect buildings. In faster RCNN the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using an RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. Once identified, its corresponding bounding box coordinates are retrieved.

Based on those coordinates, cropping task is performed. After which we perform preprocessing on the cropped images. Now, noise is removed by eliminating background details like the sky, etc. Then, we calculate the height and width of each tree texture. Then, open Blender and create a plane and resize the plane using the calculated structured information. Material and apply texture is added to the plane. Tune parameters like alpha etc. Afterward, perform UV mapping ("U" and "V" are the names of the axes of a plane) if necessary. The output now generated is a 3D tree. Thus this can be used to update virtual city to add more realism to it.

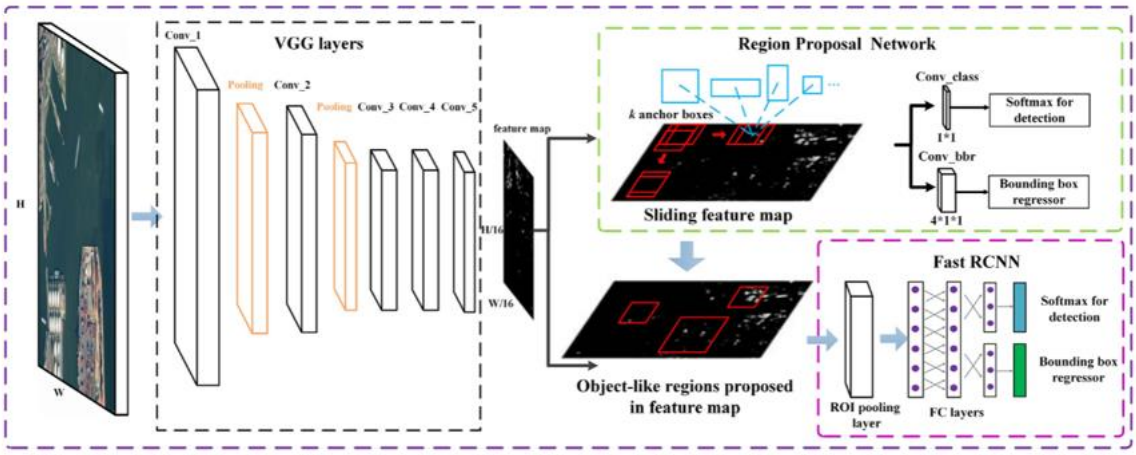


Figure 41: The architecture of Faster RCNN [68]

4.5 TRAFFIC LIGHT

Algorithm for traffic light detection

Input: Video/ Image frame

Output: Command activated using Color code

1) Perform rendering to real-time scene
2) Now, Calibrate these views
3) Perform Object detection technique like YOLO v3 to detect traffic lights
4) Once identified, its corresponding bounding box coordinates are retrieved
5) Based on those coordinates, cropping task is performed
6) Perform preprocessing on the cropped images.
7) Using OpenCV and python, retrieve the color code
8) Based on color decoded, the command will be generated for the self-driving vehicle

Flowchart for traffic light detection:

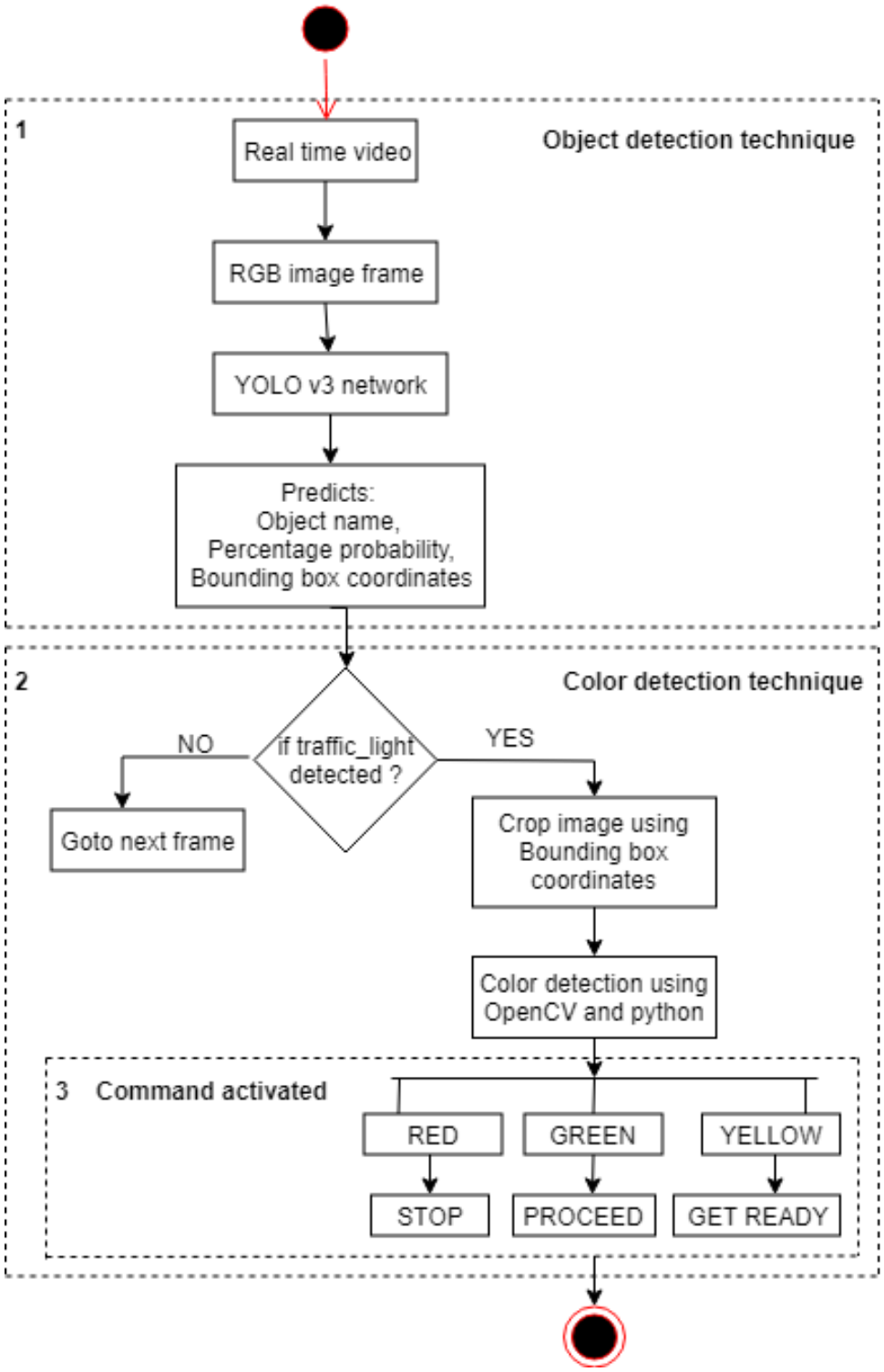


Figure 42: Flowchart for traffic light detection

Flowchart Explanation:

In the traffic light color detection module, real-time video is sent as input to the system. In the video, each frame is treated as render views. Now, calibrate these views and Perform Object detection technique like YOLO v3 to detect traffic lights. YOLO v3 uses a variant of Darknet, which initially has 53-layer network trained on Imagenet. For the object detection task, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3 [67]**.** In YOLO v3, the detection is done at three different places in the network. The first detection is done by the 82nd layer. Then, the second detection is done by the 94th layer. And third at the 106th layer.

Once traffic light is identified, its corresponding bounding box coordinates are retrieved. Based on those coordinates, cropping task is performed. Preprocessing is performed on the cropped images. Using OpenCV and python, retrieve the color code. Based on color detected, command such as STOP, PROCEED, GET READY will be activated for the self-driving vehicle to guarantee prompt car control with high accuracy to ensure safety.

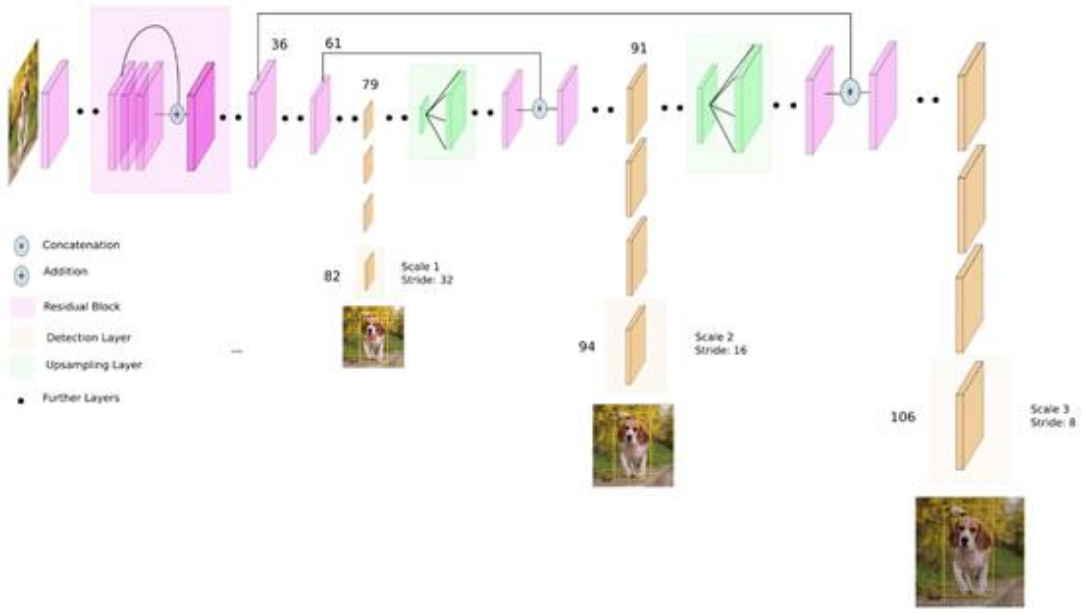


Figure 43: YOLO v3 Network Architecture [67]

4.6 Working of the overall system

The overall system consists of six modules:

1. Construction on a virtual 3D environment
2. Rendered images of real-time video
3. 3D feature and keypoint extraction
4. Removal of static and variable objects
5. Dynamic object recognition
6. Dynamic object detection

As shown in Figure 44, all these modules are interconnected which each other.

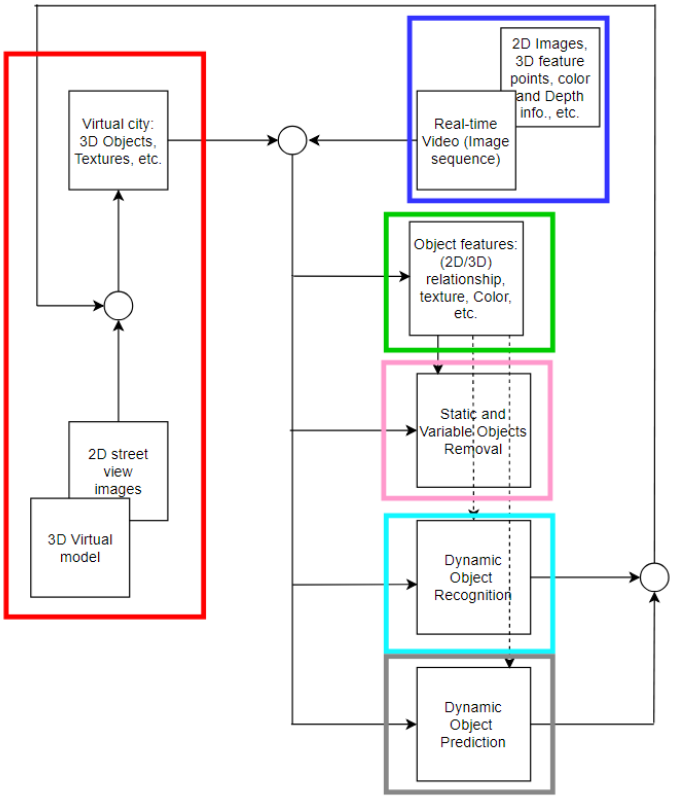


Figure 44: Overall System

In above Figure 44, work shown in the green-colored box is the contribution of this thesis work. Its connection with all other modules, which are in different colored boxes, is also depicted using arrows.

The description of the overall system is with reference to Figure 44. The system initially starts with the construction of a virtual 3D environment with the use of OpenStreetMap data and the façade texture from Google street view images. The virtual 3D city model consists of static objects, such as buildings, and some of the variable objects, such as trees. Apart from this, there is a separate repository containing 3D models of dynamic objects, such as cars is maintained. The module marked in the blue-colored box in Figure 44 shows the real-time video (image sequences) passed as input to the system. The virtual environment is rendered as corresponding to the real-time drive, and at the same time keypoint features are extracted and are stored in a repository; this work is performed in the module colored in green. The module marked in pink performs static and variable object verification and elimination by matching keypoints extracted in the previous step and key points present in its corresponding real-time image. In this module, the keypoint features of the input image (blue module) and keypoint features of the virtual environment (pink module) are matched. Matching the keypoint features of the virtual environment and real-time image confirms the location of the car in the real-world; this solves the problem of geo-localization of the self-driving car. As static and variable objects are eliminated, the computation time for the identification and prediction of dynamic objects such as human beings or animals on the road, as those are the ones which have an impact on the navigation of the self-driving system is reduced. The module marked in cyan deals with the object recognition and pose estimation of dynamic objects present in the real-time input image, such as cars. Additionally, this module tracks the recognized objects from multiple frames of the video and calculates the speed of the dynamic object. The recognized object with the pose information along with the object speed and location is used to update dynamic objects into the 3D virtual environment. The module marked in grey color updates the dynamic objects' information into the virtual environment.

4.7 Working of individual modules

The modules which are directly associated with this research work are the 3D rendering and extraction of objects features. The virtual 3D city model and the real-time video are the input to the overall system.

Creation of virtual 3D city model:

A virtual city is first created from the open-source/cloud VGI data (2D street views/satellite images). The virtual city contains the static and variable objects. Now the texture is applied to these models to resemble the real world. The constructed 3D environment is used as prior knowledge for the navigation purposes in a self-driving car.

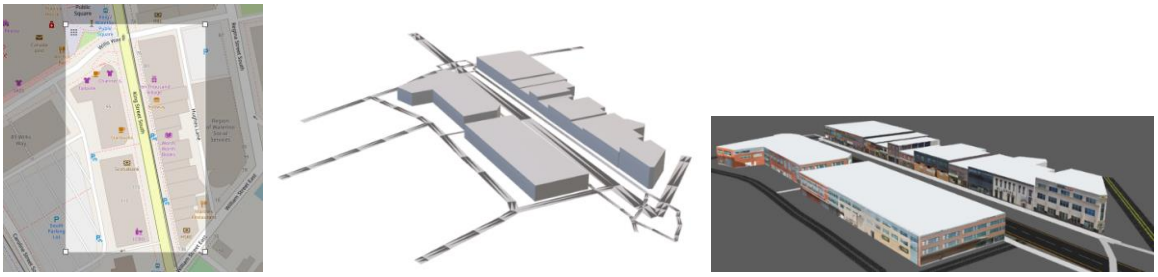


Figure 45: Pipeline for Creation of 3D virtual city

Keypoint extraction and dataset creation:

The static 3D object models are rendered and keypoint features are extracted and passed to the Static object removal module for object verification and heatmap is generated for object elimination. A repository of dynamic models is used as training set after rendering and KeypointNet [18] is used to extract the keypoint features from different rendered views of car models. The coordinate information of the identified key points in the rendered image and orientation details of the keypoint is stored in an annotation file for dynamic object recognition and detection module (will be explained in detail in Results chapter).

Static and variable object removal:

In this module, static and variable objects in the real-time image are detected with the Faster R-CNN approach. Once detected, the keypoint features of a virtual image are matched with the keypoint features of a real-time image to verify static and variable objects. Once verified, a heat map is generated for the verified images, and with the help of the heat map and the contour detection, verified objects are eliminated from the image. In this way, static

and variable objects are eliminated. This would save a lot of processing time on the dynamic object recognition in a real scene.

Figure 46: Static object elimination

Dynamic object recognition:

This module matches keypoint features of the dynamic objects in the input image with the keypoint feature information of 3D object models stored in the repository to find a suitable matching 3D model. A voting algorithm is used for the matching purpose, which also estimates a confidence score that signifies the confidence of object identification. This process improves the confidence of recognition and pose estimation of dynamic objects in the input image to update the virtual city. This module is marked in cyan color in Fig 44.

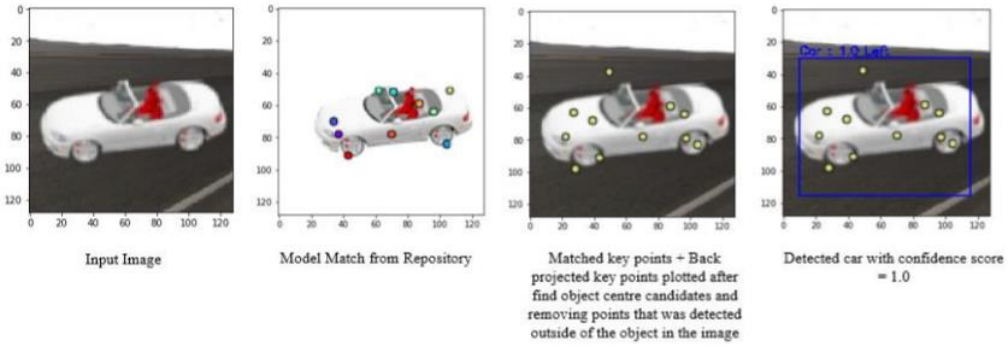


Figure 47: Dynamic object recognition and identification

Dynamic object prediction:

This module uses the information from the dynamic object recognition module to update the virtual city with dynamic objects on the road in real-time. The recognized object, with its pose information, speed and location, is used to update the virtual city with the identified

dynamic objects in real-time. Prior knowledge about the dynamic, static and variable objects from the virtual city and Internet of Things (IoT) is then used to determine the appropriate navigation decision of the self-driving car. This module is marked in grey color in Figure 44. The output of different modules is visualized in this module with the use of AirSim Simulator. Also, future prediction about the navigation of dynamic objects is calculated in this module. Moreover, this module predicts the distance from dynamic objects and visualizes them with different indicators.

Figure 48: Virtual city update and dynamic object masking

CHAPTER 5

IMPLEMENTATION AND EXPERIMENTS

The proposed approach is implemented on Windows OS using Python Programming Language. In the implementation, Python, OpenCV and other libraries were used. The list of software and tools used are mentioned in Table 2.

5.1 Software and Hardware Requirement

The implementation of proposed methodology was performed on Processor: Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz, 3301 Mhz, 6 Core(s), 12 Logical Processor(s)

Hardware	1 GPU
Operating System	Windows 10 (64-bit)
Rendering Software	Blender 2.79 Audodesk Maya 2018
Graphical User Interface	Anaconda Navigator
Programming Languages	Python 3.7.1
Markup language	XML (eXtensible Markup Language)
Integrated Development Environment	Jupyter Notebook 5.5.0
Source code editor	Notepad++
Libraries	OpenCV, TensorFlow, Keras, PIL, numpy, TensorFlow, TensorFlow-Slim, plotly, etc
3D object viewer	Microsoft 3D viewer

Table 2: List of tools used for implementation and experiments

5.2 Car

Feature Extraction Module

Incurred Preprocessing:

Rendering of 3D Model:

The image formation is the process by which a 3D representation of a scene is reduced to a 2D representation of that same scene, an image (3D scene→ transformation→ 2D image). Figure 49 below illustrates the Image Formation: Pinhole model [65].

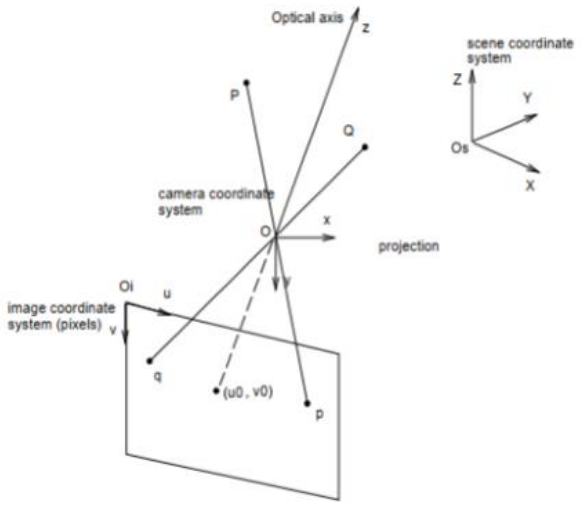


Figure 49: Image Formation: Pinhole model (Perspective model)[66]

An image point (pixel), given by its image coordinates, is the result of a three-step transformation of a physical point defined in a scene reference frame.

The following three steps are applied in sequential order: 3D Euclidean Transformation, 3D-2D transformation, 2D-2D transformation.

A 3D Euclidean Transformation:

3D rigid displacement where a scene point, initially defined in the scene reference frame, is transformed so that they would be defined in the camera reference frame. This transformation has 6 parameters corresponding to a 3D rotation and 3D translation.

A 3D-2D Transformation:

3D points defined in the camera reference frame are projected onto the image plane. These new points are called normalized coordinates.

A 2D-2D Transformation:

The normalized coordinates expressed in the scene metrics, undergo a 2D affine transformation to become defined in pixels in the image plane reference frame.

After the three steps, a perspective projection of the 3D point, P = (X, Y, Z, 1) onto the pixel p = (u, v, 1) is done using the projection matrix information. u and v can be defined using below equations [65]:

$$u = \frac{m11X+m12Y+m13Z+m14}{m31X+m32Y+m33Z+m34}$$

$$v = \frac{m21X+m22Y+m23Z+m24}{m31X+m32Y+m33Z+m34}$$

It is clear from the above equation of u and v that, u and v coordinates make use of the X, Y, Z information of the image plane onto the pixel, thus saving the depth information.

Concept Involved:

Given a single 2D image of a known class, Keypoint Network can predict a set of 3D key points that are consistent across viewing angles of the same object and across object instances. These key points and their detectors are discovered and learned automatically without keypoint location supervision [18].

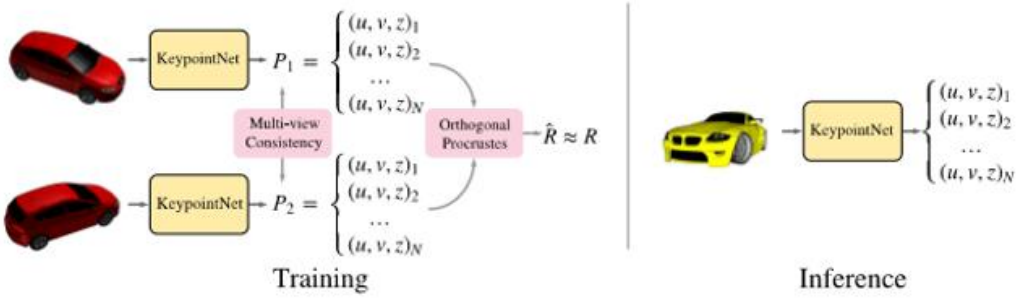


Figure 50: Training and Inference phase using Keypointnet [18]

Dataset Used:

We use ShapeNet dataset for our training and testing datasets. The ShapeNetCore subset of ShapeNet contains about 51,300 3D models over 55 common categories, each subdivided into several subcategories [12].

Input:

185 3D car models are represented as a series of 2D images that are obtained from different views on the model. For each model, we normalize the object so that the longest dimension lies in [−1,1], and render 240 images of size 128×128 under different viewpoints. The camera viewpoints are randomly sampled around the object from a fixed distance, all above the ground with zero roll angle. We then add small random shifts to the camera positions.

Implementation details:

We implemented our network in TensorFlow and trained with Adam optimizer with a learning rate of 10^{-3}, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and a total batch size of 256. We use the following weights for the losses: (α_{con}, α_{pose}, α_{sep}, α_{obj}) = (1,0.2,1.0,1.0). We train the network for 200K steps using synchronous training with 32 replicas. All kernels for all layers are 3×3, and 13 layers of dilated convolutions are stacked with dilation rates of 1, 1, 2, 4, 8, 16, 1, 2, 4, 8, 16, 1, 1, all with 64 output channels except for the last layer which has 2N output channels, split between probability distribution map generation and depth value prediction. LeakyRelu and Batch Normalization is used for all layers except for the last layer. The output layers for depth value prediction have no activation function, and the channels are passed through a spatial softmax to produce a probability distribution map. Finally, generated probability distribution map and predicted depth value is then converted to actual coordinates using certain equations [18].

Training hours:

Training almost took 20 days to generate KeypointNet.

Output:

Each image produces 10 salient points. Thus, each 3D car model is described by a set of 2.4k keypoints and Orientation information of each key point. Save them in the repository which can be used for feature matching and object identification.

Result:

10 key points, corresponding orientation and spatial transformation are shown for image 000001 in Figure 43.

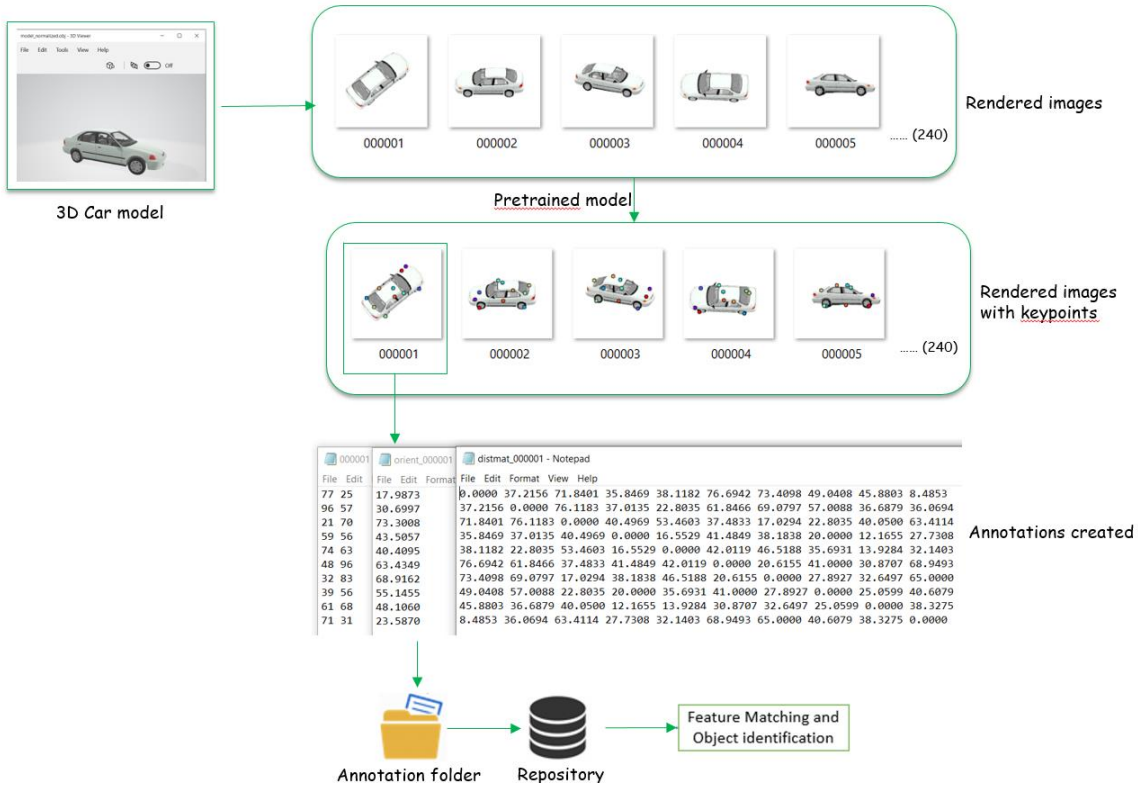


Figure 51: Car feature extraction result

5.3 Building

Feature Extraction Module

Incurred Preprocessing:

Inspired by the success of Faster R-CNN in both detection accuracy and detection speed, this work proposed an object detection method based on Faster R-CNN to detect buildings from the virtual world.

Concept Involved:

With FAST, corner detection was prioritized over edges as they claimed that corners are one of the most intuitive kinds of characteristics showing a sharp two-dimensional shift in intensity and are thus well-differentiated from adjacent points. FAST uses 16 pixels circle to evaluate whether point p is a corner or not.

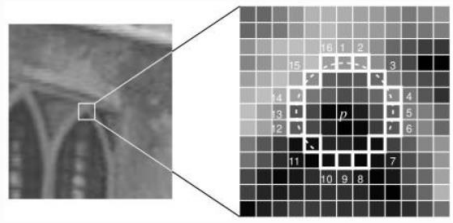


Figure 52: 16 pixels circle involved in the FAST algorithm[70]

Dataset used:

The textured 3D virtual model that we used in our work corresponds to the region of King St S at Wills Way to King St S at William St E, Waterloo, Ontario. But for implementing Faster RCNN, training model is built using google open images dataset.

Input:

The video contains a sequence of 2D rendered images that are obtained by moving the virtual camera across the lane in the virtual world. Next, break it into image frames.

Implementation details:

The first step includes extracting sub-data from Open Images Dataset V4 which includes downloading the images and creating the annotation files for our training. The second part is to train the model. The configuration and model saved path are inside this file. Final Step of object detection is to test the model with test images and calculate the mAP (mean average precision) for the model. Coming to the feature extraction part, a FAST algorithm is applied along with the non-maximum suppression step to fetch key points. To conclude, selective features needs to be far from the center of the image and are obtained by calculating the Euclidean distance between detected points and the center of the image.

Training hours:

It almost took 3 days to train the model for object detection using Faster RCNN.

Output:

The extracted features are used to build selective features by fetching extreme 'n'(say n=32) points detected on image using Euclidean distance formula. Save them in the repository for feature matching and object verification.

Results:

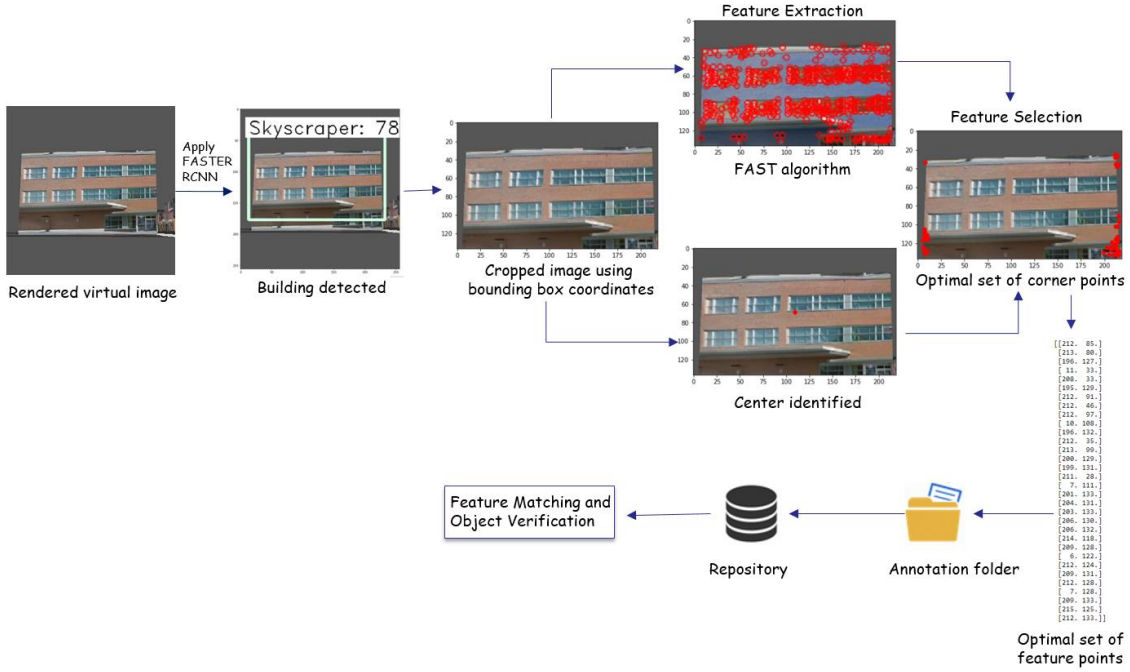


Figure 53: Building feature extraction result

Building heatmap generation module

Incurred Preprocessing:

Inspired by the success of Faster R-CNN in both detection accuracy and detection speed, this work proposed an object detection method based on Faster R-CNN to detect buildings from the virtual world.

Concept Involved:

A heatmap is a 2D visual/graphical representation of data where the individual values are represented as colors that are contained in a matrix. Heatmaps are used in various analysis but we are using if for eliminating pixels with some threshold value.

Dataset Used:

The real-time video/images that we used in our work corresponds to the region of King St S at Wills Way to King St S at William St E, Waterloo, Ontario. But for implementing Faster RCNN, training model is built using google open images dataset.

Input:

The video contains a sequence of 2D rendered images that are obtained by moving the embedded camera across the lane in real-world. Next, break it into image frames.

Implementation details:

Once cropped image obtained using Faster-RCNN, remove noise by eliminating background details like the sky, etc. Then, execute applyColorMap() to its grayscaled image. A colormap is a mapping from 0-255 values to 256 colors. In OpenCV, we can create an 8-bit color image of size 256 x 1 to store the 256 color values. Now generate heatmap by tuning parameters.

Training Hours:

It almost took 3 days to train the model for object detection using Faster RCNN.

Output:

Generates heatmap which is called as binary heatmap as it contains only binary digits. Finally, save binary heatmap in the repository for object elimination.

Result:

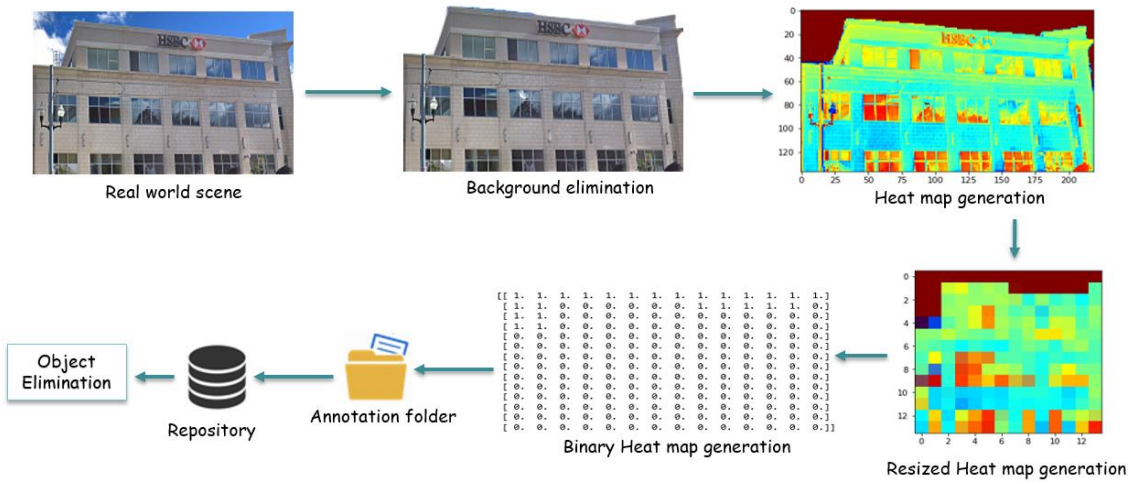


Figure 54: Building heatmap generation

5.4 Trees

3D reconstruction module

Incurred Preprocessing:

Inspired by the success of Faster R-CNN in both detection accuracy and detection speed, this work proposed an object detection method based on Faster R-CNN to detect trees from the real world.

Concept Involved:

For architectural visualization, a scene produced is never complete without some components or objects such as trees that represent a real sense of scale for the viewer. Blender 3D has a few options to add those type of elements to a scene, going from a 3D model of trees to the use of 2D images.

Dataset Used:

The real-time video/images that we used in our work corresponds to the region of King St S at Wills Way to King St S at William St E, Waterloo, Ontario. But for implementing Faster RCNN, training model is built using google open images dataset.

Input:

The video contains the sequence of 2D rendered images that are obtained by moving the embedded camera across the lane in the real world. Next, break it into image frames and perform object detection using Faster RCNN.

Implementation details:

With the cropped image saved to .png, leaving the background transparent. Open Blender and create a plane. Scale the plane and add material and an image texture to the plane. Next, go to the materials panel and set the alpha of the material as 0, Ztransp as transparency enabled and make shadeless so as to make the material insensitive to light or shadow. Use UV Mapping to control and see the texture in 3D View.

Training Hours:

It almost took 3 days to train the model for object detection using Faster RCNN.

Output:

In the end, 3D models of trees are produced which can be used to add realism in the 3D virtual city.

Result:

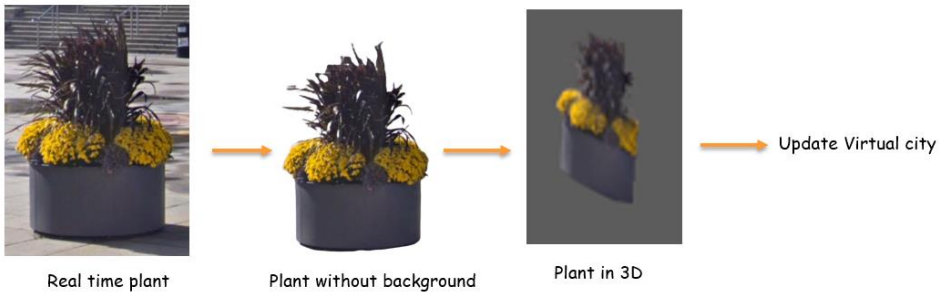


Figure 55: Tree 3D reconstruction

5.5 Traffic light detection module

Incurred Preprocessing:

YOLO v3 is used to detect traffic light from the real world.

Concept Involved:

Object detection technique like YOLO v3 to detect traffic lights. YOLO v3 uses a variant of Darknet, which initially has 53-layer network trained on Imagenet. For the object detection task, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3. It contains a high number of annotated traffic lights, captured in varying light and weather conditions.

Dataset Used:

The real-time video/images that we used in our work corresponds to the region of King St S at Wills Way to King St S at William St E, Waterloo, Ontario. But for implementing Faster RCNN, training model is built using google open images dataset.

Input:

The video contains the sequence of 2D rendered images that are obtained by moving the embedded camera across the lane in the real world. Next, break it into image frames and perform object detection using YOLO v3.

Implementation details:

Once YOLO v3 is applied, crop the image using its bounding boxes. Send the cropped image to color detection code to detect the color pixel which occupies the major portion.

Training Hours:

Used Pre-trained model: YOLO v3

Output:

Based on color detected, command such as STOP, PROCEED, GET READY will be activated for the self-driving vehicle to guarantee prompt car control with high accuracy to ensure safety.

Result:

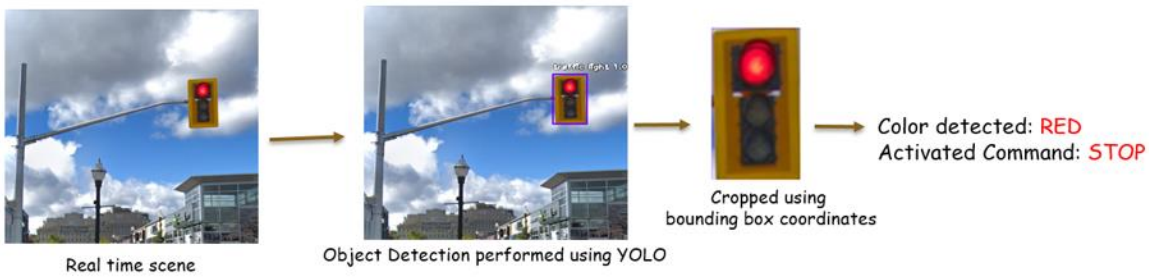


Figure 56: Traffic light color detection module

CHAPTER 6

RESULTS AND EVALUATION

6.1 Car: Graphs generated using training data

Figure 57 represents the variation in accuracy w.r.t increase in epoch value obtained from training the car dataset using Dilated CNN.

Figure 57: epoch vs accuracy in car's training

Figure 58 depicts the variation in total loss w.r.t increase in epoch value obtained from training the car dataset using Dilated CNN.

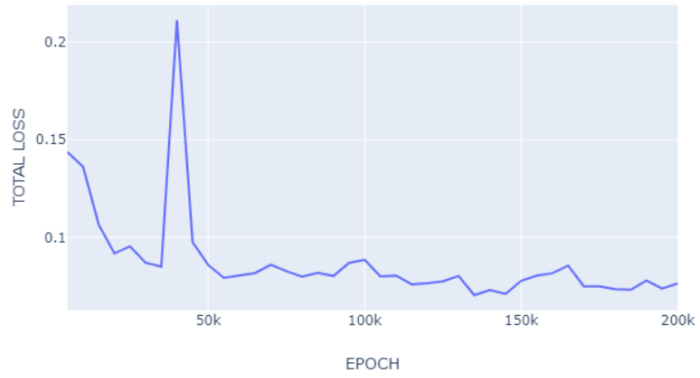


Figure 58: epoch vs total loss in car's training

6.2 Buildings and Trees: Graphs generated using training data

Figure 59 illustrates the variation in the mean of overlapping bounding boxes w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

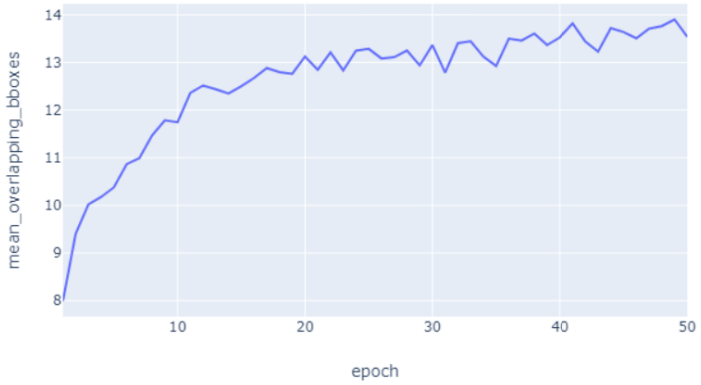


Figure 59: epoch vs mean_overlapping_bboxes

Figure 60 shows the variation in the class accuracy w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

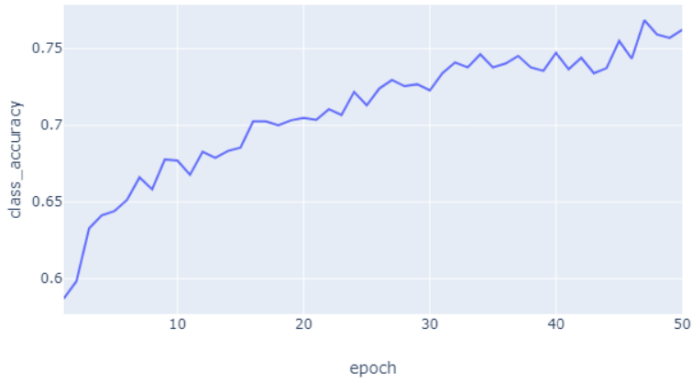


Figure 60: epoch vs class_accuracy

Figure 61 portrays the variation in the loss in RPN classification w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

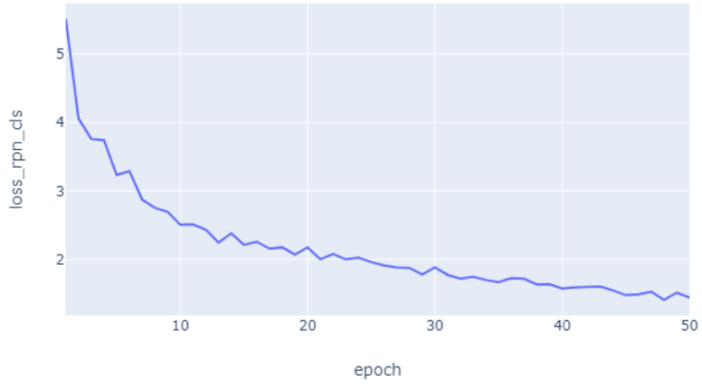


Figure 61: epoch vs loss_rpn_cls

Figure 62 interprets the variation in the loss in RPN regression w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

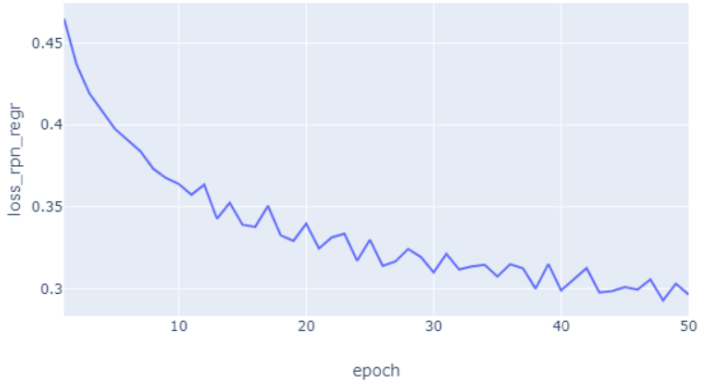


Figure 62: epoch vs loss_rpn_regr

Figure 63 depicts the variation in the current loss w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

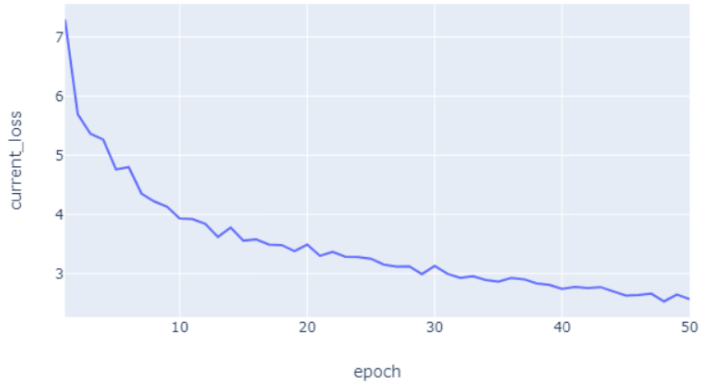


Figure 63: epoch vs current_loss

Figure 64 describes the variation in the elapsed time w.r.t increase in epoch value obtained from training the building and trees dataset using Faster RCNN.

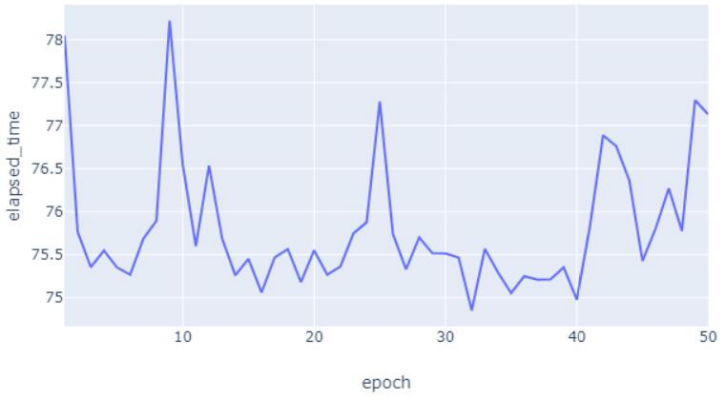


Figure 64: epoch vs elapsed_time

6.3 Performance Measures (for cars):

- **Multi-view consistency loss (L_{con}):**

Multi-view consistency loss measures the discrepancy between the two sets of points under the ground truth transformation.

$$L_{\text{con}} = \frac{1}{2N} \sum_{i=1}^{N} \left\| [u_i, v_i, u_i', v_i']^\top - [\hat{u_i'}, \hat{v_i'}, \hat{u_i}, \hat{v_i}]^\top \right\|^2$$

[x,y,z] to denote 3D coordinates, and [u,v] to denote pixel coordinates. ŭ denotes the projection of u to the second view, and u'^ denotes the projection of ŭ to the first view.

- **A relative pose estimation loss (L_{pose}):**

A relative pose estimation loss, which penalizes the angular difference between the ground truth rotation R and the rotation Ř recovered from P1 and P2 (two views that best match one view to the other) using orthogonal procrustes.

$$L_{\text{pose}} = 2 \arcsin\left(\frac{1}{2\sqrt{2}} \left\| \hat{R} - R \right\|_F\right)$$

- **Total loss** = multi-view consistency loss + relative pose estimation loss

6.4 Evaluations and Comparisons

Car

AUTHOR NAME	Models	Learning	Drawbacks	Annotations
Suwajanakorn, S., et al. 2018 [18]	Train set = 149 Test set =18 Val test =18	Unsupervised	The Keypointnet orientation network fails to predict correct orientation and the output keypoints are flipped in the case of cars whose front and back look similar. Keypoints are calculated but no additional information regarding the key points is provided like orientation, spatial transformations, etc	No
Our Approach	Train set =149 Test set =18 Val test =18	Unsupervised	The Keypointnet orientation network fails to predict correct orientation and the output keypoints are flipped in the case of cars whose front and back look similar	Yes (-2D key points -orientation of the keypoints -spatial transformations of the key points)

Table 3: Suwajanakorn, S., et al. 2018 vs Our approach

	SIFT	HOG 2d/3d	Ours
Keypoint Orientation formula	$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$ $\theta(x,y) = tan^{-1}(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)})$	$\begin{cases} m(x,y) & = \sqrt{L_x^2 + L_y^2} \\ \theta(x,y) & = \tan^{-1}(\frac{L_y}{L_x}) \end{cases}$ $\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} \arccos \frac{z}{\sqrt{x^2+y^2+z^2}} \\ \text{atan2}(x,y) \\ \sqrt{x^2+y^2+z^2} \end{pmatrix}$	$m(x,y)=\sqrt{x^2+y^2}$ $\Theta(x,y) = \tan^{-1}(y/x)$

Table 4: Comparison between different Keypoint Orientation formulae

Buildings

Various feature extraction (corner detection) technique are tested with our virtual images

IMAGES	FASTER RCNN ACCURACY(%)		FORSTNER	SHI AND TOMASI	HARRIS	FAST
1	78	dp	93	113	150	526
		t	3.478355646133423	0.15987443923950195	0.15987420082092285	0.18085908889770508
		dp/t	26.7367714694	706.804668323	938.237684566	2908.34153376
2	70	dp	117	135	178	644
		t	4.066396713256836	0.14788508415222168	0.17138099670410156	0.21083498001098633
		dp/t	28.7724017725	912.870968522	1038.62157079	3054.52159773
3	77	dp	39	68	96	362
		t	1.8705601692199707	0.15838313102722168	0.12541460990905762	0.17686057090759277
		dp/t	20.8493694251	429.338652159	765.461058083	2046.81008402
4	61	dp	38	52	177	320
		t	1.7251849174499512	0.15188264846801758	0.160874605178833	0.16187381744384766
		dp/t	22.0266242857	342.369589446	1100.23580044	1976.84841844
5	89	dp	54	69	96	473
		t	1.9190306663513184	0.1563887596130371	0.12490320205688477	0.16986393928527832
		dp/t	28.1392063956	441.208179992	768.595187466	2784.58160096

6	77	dp	57	71	123	575
		t	2.0269505977630615	0.16986823081970215	0.1568775177001953	0.1898508071899414
		dp/t	28.1210603075	417.971033532	784.051161716	3028.69399667
7	78	dp	62	82	137	509
		t	2.4211206436157227	0.17486357688903809	0.1563870906829834	0.16886687278747559
		dp/t	25.6079762747	468.936993391	876.031387256	3014.20871719
8	72	dp	39	61	150	437
		t	1.7006986141204834	0.15787601470947266	0.1558775901794433	0.1608738899230957
		dp/t	22.9317526787	386.379147664	962.293552443	2716.41346031
9	80	dp	38	53	164	232
		t	1.7551617622375488	0.15038704872131348	0.1603829860687255	0.1338942050933838
		dp/t	21.6504260847	352.423965033	1022.55235434	1732.71128379
10	83	dp	58	64	109	342
		t	2.0469346046447754	0.15738797187805176	0.1433947086334228	0.1558771133422851 6
		dp/t	28.3350527508	406.638444071	760.139624668	2194.03601123

Table 5: Various corner detection techniques tested with our virtual images

dp - number of detected points, **t** - time taken for detecting points (sec)
dp/t - detection rate, **Accuracy** - Accuracy generated by Faster RCNN
green - highest value, **orange** - least value

6.5 Advantages of proposed methodology

- It adds realism to virtual city by updating it with 3D tree models.
- The feature points generated for virtual building models is used for feature matching and object verification for elimination.
- The heat map generated on the real time building images is used for object elimination so as to detect dynamic objects in less time.

- Using Faster RCNN for training trees and buildings, we achieved high mean Average Precison value as 0.765.

6.5.1 Advantage to dependency modules

Based on buildings - Object elimination:

- After applying the proposed approach of masking, the building will be eliminated, allowing dynamic object detection algorithm to focus on actual dynamic objects
- The running time of object elimination algorithm is 548 milliseconds, and dynamic object detection algorithm is 15.8 seconds on masked image
- The proposed approach makes moving object detection method to perform efficiently and accurately

Based on cars - Feature extraction:

- Improvement in confidence of object recognition
  Based on this sequence flow: mAP = 91.0
  whereas Tangruamsub et al. 2011 stated only 86
- Pose estimation also improved
- No incorrect detection of objects by the system

6.6 Limitations

- FAST algorithm does not detect corners on computer-generated images that are perfectly aligned to the x-axes and y-axes.
- However, Faster R-CNN pipelines are fitting for object detection in still images. When these methods are applied to videos, they might miss some positive samples because the objects might not be of their best poses in each frame of the videos.
- The Keypointnet orientation network fails to predict correct orientation and the output keypoints are flipped in the case of cars whose front and back look similar.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

The concept of the proposed approach takes advantage of a priori source of information to perform feature extraction and generate heat maps on real-time building objects. Our dataset consists of selective features from a virtual city. We have conducted a wide range of experiments and provided a comprehensive analysis of the performance of the FAST algorithm against other techniques. Ultimately, these results motivate future research on synchronous learning from real and virtual data. The experimental results also show that our framework is robust for the extraction of features of buildings and cars with complex shapes. Overall, we can affirm that we have driven along the route of research for training our car to see using virtual worlds.

7.2 Future Works

The research enables many more opportunities to improve further. The number of test images can be increased substantially and tested with streaming videos to perform more practical real-world applications of self-driving cars and other autonomous vehicles. The performance of mAP scores and inference time can be improved by tuning other hyper-parameters using high-performance GPU systems. In the future, the entire system compatibility will be improved in such a way that a hardware implementation will be done. Scalability issues require further attention to extend the proposed approach to city size maps. The algorithms which will be in fruition in the near future will be mostly aimed towards reducing such complexities, increasing the performance as well as finding ways to minimize data dependency to make more efficient and practical systems.

REFERENCES/BIBLIOGRAPHY

1) Simhambhatla, R., Okiah, K., Kuchkula, S., & Slater, R. (2019). Self-Driving Cars: Evaluation of Deep Learning Techniques for Object Detection in Different Driving Conditions. *SMU Data Science Review*, *2*(1), 23.
2) https://www.digitalfife.com/webs/83/documents/WhattheTech.pdf
3) https://en.wikipedia.org/wiki/3D_computer_graphic
4) Cappelle, C., El Najjar, M. E., Charpillet, F., & Pomorski, D. (2012). Virtual 3D city model for navigation in urban areas. *Journal of Intelligent & Robotic Systems*, *66*(3), 377-399.
5) Salahat, E., & Qasaimeh, M. (2017, March). Recent advances in features extraction and description algorithms: A comprehensive survey. In *2017 IEEE international conference on industrial technology (ICIT)* (pp. 1059-1063). IEEE.
6) Karim, S., Zhang, Y., Asif, M. R., & Ali, S. (2017). Comparative analysis of feature extraction methods in satellite imagery. *Journal of Applied Remote Sensing*, *11*(4), 042618.
7) El-Gayar, M. M., & Soliman, H. (2013). A comparative study of image low level feature extraction algorithms. *Egyptian Informatics Journal*, *14*(2), 175-181.
8) Dawood, M., Cappelle, C., El Najjar, M. E., Khalil, M., & Pomorski, D. (2012, October). Harris, SIFT and SURF features comparison for vehicle localization based on virtual 3D model and camera. In *2012 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA)* (pp. 307-312). IEEE.
9) Rajam, F., & Valli, S. (2013). A survey on content based image retrieval. *Life Science Journal*, *10*(2), 2475-2487.
10) Gabriel, E. (2019). *Automatic Multi-Scale and Multi-Object Pedestrian and Car Detection in Digital Images Based on the Discriminative Generalized Hough Transform and Deep Convolutional Neural Networks* (Doctoral dissertation, Christian-Albrechts Universität Kiel).
11) https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2018/01/Uber-self-driving-car-796x419.jpg

12) Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., ... & Xiao, J. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

13) Joshi, P., & Sehra, C. (2018). Object Detection Algorithms: A Brief Overview.

14) https://interestingengineering.com/what-is-deep-learning-and-why-is-it-more-relevant-than-ever

15) https://webdocs.ca/deeplearning

16) Huang, L., Yang, Y., Deng, Y., & Yu, Y. (2015). Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*.

17) Xu, Y., Yu, G., Wang, Y., Wu, X., & Ma, Y. (2017). Car detection from low-altitude UAV imagery with the faster R-CNN. *Journal of Advanced Transportation*, *2017*.

18) Suwajanakorn, S., Snavely, N., Tompson, J. J., & Norouzi, M. (2018). Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems* (pp. 2059-2070).

19) Aubry, M., & Russell, B. C. (2015). Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2875-2883).

20) R. Girshick, "Fast R-CNN," in Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15), pp. 1440–1448, December 2015.

21) Li, Y., Zhang, X., & Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1091-1100).

22) Li, J., Yu, Z., Gu, Z., Liu, H., & Li, Y. (2019). Dilated-Inception Net: Multi-Scale Feature Aggregation for Cardiac Right Ventricle Segmentation. *IEEE Transactions on Biomedical Engineering*.

23) Sadigh, S., & Sen, P. (2018). Improving the Resolution of CNN Feature Maps Efficiently with Multisampling. *arXiv preprint arXiv:1805.10766*.

24) Yang, T., Qin, S., Yan, J., & Zhang, W. (2018, July). Multi-Label Dilated Recurrent Network for Sequential Face Alignment. In *2018 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1-6). IEEE.

25) Wang, T., Sun, M., & Hu, K. (2017, November). Dilated deep residual network for image denoising. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 1272-1279). IEEE.

26) Wang, J. H., Lin, G. F., Chang, M. J., Huang, I. H., & Chen, Y. R. (2019). Real-Time Water-Level Forecasting Using Dilated Causal Convolutional Neural Networks. *Water Resources Management*, 1-22.

27) Lal, K., & Arif, K. M. (2016, August). Feature extraction for moving object detection in a non-stationary background. In *2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)* (pp. 1-6). IEEE.

28) https://distill.pub/2018/building-blocks/

29) https://www.researchgate.net/publication/317887580_Performance_Analysis_of_Corner_Detector_Algorithm_in_Image_Noise_Operator_Comparison_Study

30) Kaneva, B., Torralba, A., & Freeman, W. T. (2011, November). Evaluation of image features using a photorealistic virtual world. In *2011 International Conference on Computer Vision* (pp. 2282-2289). IEEE.

31) https://www.quora.com/What-is-the-difference-between-dilated-convolution-and-convolution+stride

32) Tang, S., Li, Y., Deng, L., & Zhang, Y. (2017). Object localization based on proposal fusion. *IEEE Transactions on Multimedia*, *19*(9), 2105-2116.

33) Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

34) Zhou, Y., Liu, L., & Shao, L. (2018). Vehicle re-identification by deep hidden multi-view inference. *IEEE Transactions on Image Processing*, *27*(7), 3275-3287.

35) Su, H., Qi, C. R., Li, Y., & Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2686-2694).

36) Li, J., & Allinson, N. M. (2008). A comprehensive review of current local features for computer vision. *Neurocomputing*, *71*(10-12), 1771-1787.Kulyk, V., & Sossa, R. (2018).

37) Determining the tourist attractive regions by GIS analysis using heatmaps. *Geodesy and Cartography*, *44*(1), 22-27.

38) analysis using heatmaps. *Geodesy and Cartography*, *44*(1), 22-27.

39) Roa Rodríguez, R., & Lundin, R. (2016). Heatmap Visualization of Neural Frequency Data.

40) Babicki, S., Arndt, D., Marcu, A., Liang, Y., Grant, J. R., Maciejewski, A., & Wishart, D. S. (2016). Heatmapper: web-enabled heat mapping for all. *Nucleic acids research*, *44*(W1), W147-W153.

41) Yamaguchi, M., Saito, H., & Yachida, S. (2017, January). Application of LSD-SLAM for Visualization Temperature in Wide-area Environment. In *VISIGRAPP (4: VISAPP)* (pp. 216-223).

42) T. Faulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N.Hawes, P. Jensfelt, and M. Vincze. Autonomous learning of object models on a mobile robot. *Robotics and Automation Letters (IEEE Journal)*, Volume 2(Issue 1):26–33, 2016.

43) W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation. *ECCV*, 9907(7):205–220, 2016.

44) E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. *ICCV*, pages 118–126, 2015.

45) Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 129-137).

46) Wang, Z., Tang, L., Liu, X., Yao, Z., Yi, S., Shao, J., ... & Wang, X. (2017). Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 379-387).

47) Chabot, F., Chaouch, M., Rabarisoa, J., Teulière, C., & Chateau, T. (2017). Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2040-2049).

48) Song, X., Wang, P., Zhou, D., Zhu, R., Guan, C., Dai, Y., ... & Yang, R. (2019). Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5452-5462).

49) http://www.researchinchina.com/UpLoads/Article/2018/RADAR%202_%E5%89%AF%E6%9C%AC.png

50) Xi, W., Shi, Z., & Li, D. (2017). Comparisons of feature extraction algorithm based on unmanned aerial vehicle image. *Open Physics*, *15*(1), 472-478.

51) Li, H., Qin, J., Xiang, X., Pan, L., Ma, W., & Xiong, N. N. (2018). An efficient image matching algorithm based on adaptive threshold and RANSAC. *IEEE Access*, *6*, 66963-66971.

52) Gao, B. B., Wei, X. S., Wu, J., & Lin, W. (2015). Deep spatial pyramid: The devil is once again in the details. *arXiv preprint arXiv:1504.05277*.

53) Warrington, A., & Wood, F. (2017). Updating the VESICLE-CNN Synapse Detector. *arXiv preprint arXiv:1710.11397*.

54) Prathap, K. S. V., Jilani, S. A. K., & Reddy, P. R. (2016, January). A critical review on Image Mosaicing. In *2016 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-8). IEEE.

55) Qureshi, F., Memon, I., Memon, F. A., Dahri, S. A., & Memon, F. (2016). A study on image detection techniques. *International Journal of Engineering Research and Development*, *12*(8), 34-39.

56) Siddiqi, M. H., Ali, R., Khan, A. M., Kim, E. S., Kim, G. J., & Lee, S. (2015). Facial expression recognition using active contour-based face detection, facial movement-based feature extraction, and non-linear feature selection. *Multimedia Systems*, *21*(6), 541-555.

57) https://www.geeksforgeeks.org/python-corner-detection-with-shi-tomasi-corner-detection-method-using-opencv/.

58) Hayat, M., Bennamoun, M., & El-Sallam, A. (2012, June). Evaluation of spatiotemporal detectors and descriptors for facial expression recognition. In *2012 5th International Conference on Human System Interactions* (pp. 43-47). IEEE.

59) Li, H., Zhao, T., Li, N., Cai, Q., & Du, J. (2017). Feature Matching of Multi-View 3D Models Based on Hash Binary Encoding. *Neural Network World*, *27*(1), 95.

60) Ohbuchi, R., & Furuya, T. (2008, December). Accelerating bag-of-features sift algorithm for 3d model retrieval. In *Proc. SAMT 2008 Workshop on Semantic 3D Media (S-3D)* (pp. 23-30).

61) Furuya, T., & Ohbuchi, R. (2009, July). Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features. In *Proceedings of the ACM international conference on image and video retrieval* (p. 26). ACM.

62) Bai, S., Bai, X., Zhou, Z., Zhang, Z., & Jan Latecki, L. (2016). Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5023-5032).

63) Khan, S. D. (2014). Estimating Speeds and Directions of Pedestrians in Real-Time Videos: A solution to Road-Safety Problem. In *CEUR Workshop Proceedings* (p. 1122).

64) Jensen, M. B., Nasrollahi, K., & Moeslund, T. B. (2017). Evaluating state-of-the-art object detector on challenging traffic light data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 9-15).

65) Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

66) B.Boufama, "Image Formation and Camera Model", Lecture Notes, University of Windsor.

67) https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

68) Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., & Zou, H. (2018). Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS journal of photogrammetry and remote sensing*, *145*, 3-22.

69) https://ichef.bbci.co.uk/news/660/cpsprodpb/B6A6/production/_87685764_1c5784b9-2f8d-44d5-bbd6-bac13893236f.jpg

70) https://opencv-python tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html

71) Dollár, Piotr and Zitnick, C Lawrence. “Fast Edge Detection Using Structured Forests”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 37.8 (2015), pp. 1558–1570.

72) http://stanislaschaillou.com/suggestive-cad/IMG/precedents_1.jpg

73) Benuwa, B. B., Zhan, Y. Z., Ghansah, B., Wornyo, D. K., & Banaseka Kataka, F. (2016). A review of deep machine learning. In *International Journal of Engineering Research in Africa* (Vol. 24, pp. 124-136). Trans Tech Publications.

74) https://player.slideplayer.com/85/13737497/slides/slide_14.jpg

75) Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

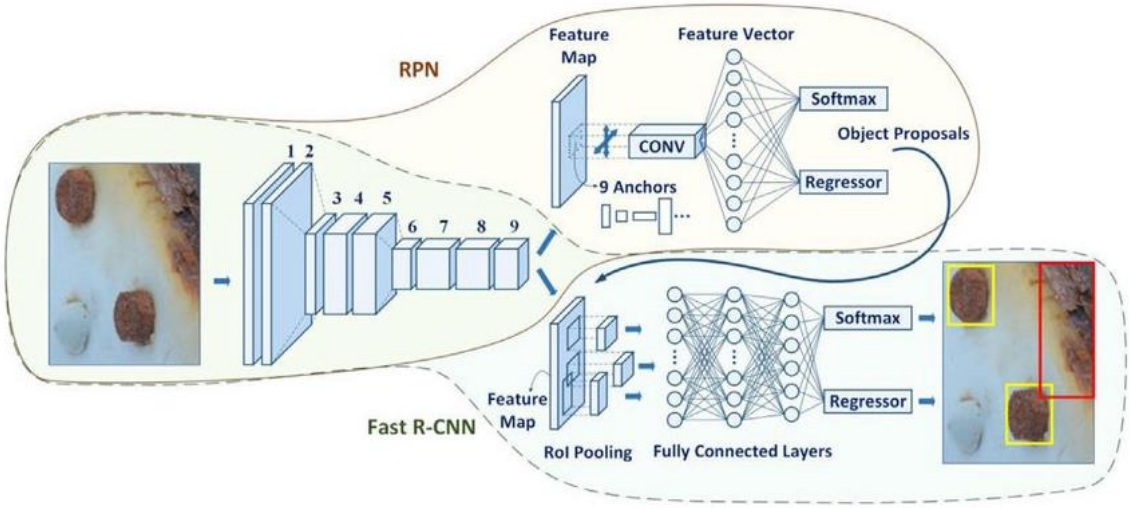
76) Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).

APPENDIX

Faster RCNN:

Faster RCNN is an object detection architecture presented by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015, and is one of the famous object detection architectures that uses convolution neural networks like YOLO (You Look Only Once) and SSD (Single Shot Detector).

Faster RCNN is composed from 3 parts



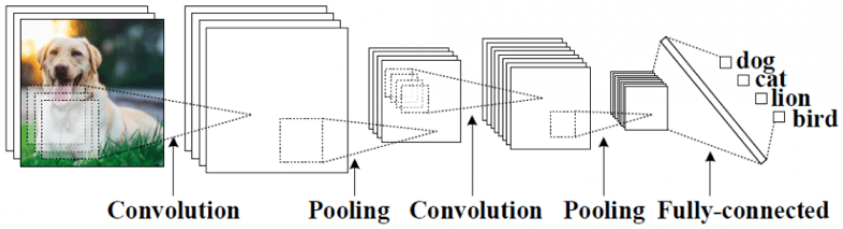
- **Part 1: Convolution layers**

In this layers we train filters to extract the appropriate features the image, for example let's say that we are going to train those filters to extract the appropriate features for a human face, then those filters are going to learn through training shapes and colors that only exist in the human face.

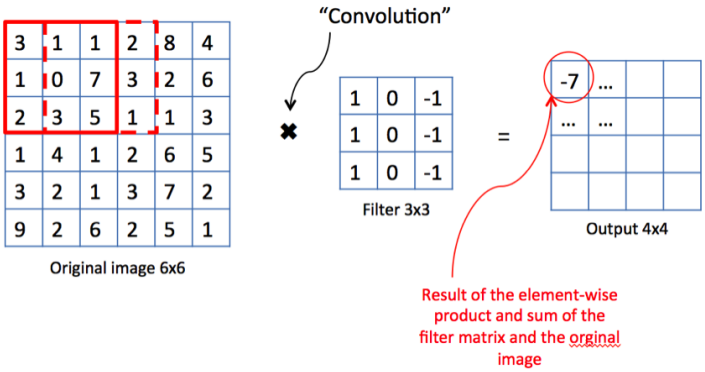
So we can assimilate convolution layers to coffee filters, coffee filter doesn't let the coffee powder pass to the cup so our convolutions layer that learn the object features and don't let anything else pass, only the desired object.

- Coffee powder + Coffee liquid = Input image
- Coffee filter = CNN filters
- Coffee liquid = Last feature map of the CNN

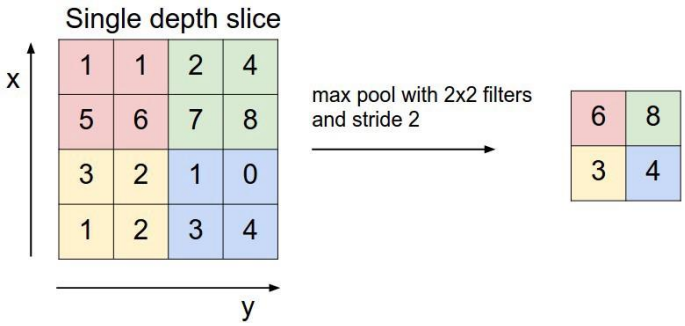
Convolution networks are generally composed of Convolution layers, pooling layers and a last component which is the fully connected or another extended thing that will be used for an appropriate task like classification or detection.



We compute convolution by sliding filter all along our input image and the result is a two-dimension matrix called feature map.



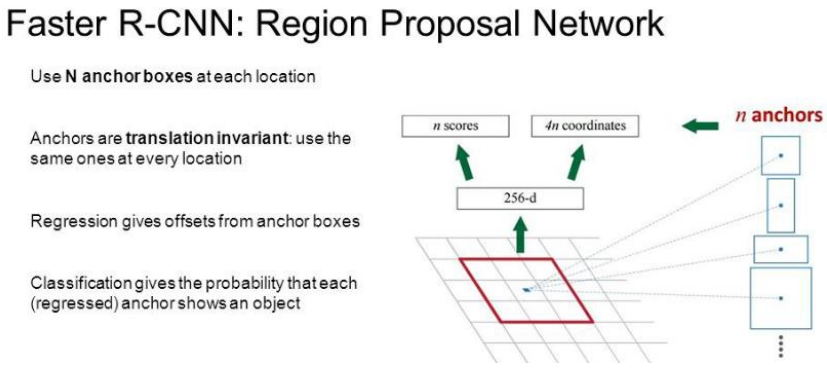
Pooling consists of decreasing quantity of features in the features map by eliminating pixels with low values.



And the last thing is using the fully connected layer to classify those features which not our case in the Faster RCNN.

- **Part 2: Region Proposal Network (RPN)**

RPN is small neural network sliding on the last feature map of the convolution layers and predict whether there is an object or not and also predict the bounding box of those objects.



- **Part 3: Classes and Bounding Boxes prediction**

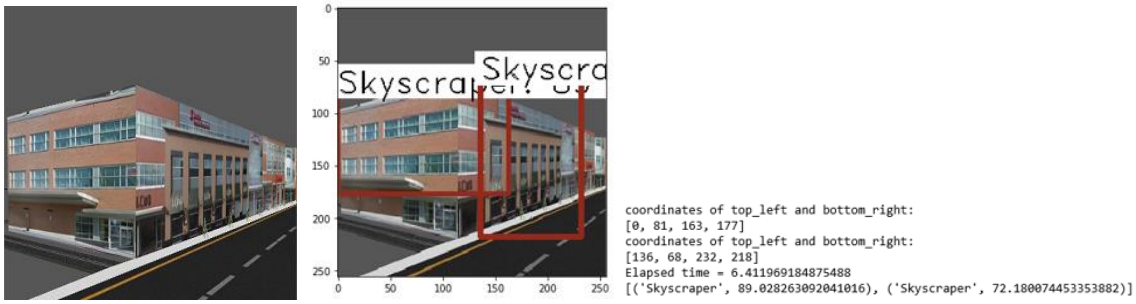
Now we use another Fully connected neural networks that takes as an input the regions proposed by the RPN and predict object class (classification) and Bounding boxes (Regression).

- **Training**

To train this architecture, we use SGD to optimize convolution layers' filters, RPN weights and the last fully connected layer weights.

Examples:

Multiple object detection



Input image After applying faster RCNN

VITA AUCTORIS

NAME	Shivani Pachika
PLACE OF BIRTH	Telangana, India
YEAR OF BIRTH	1995
EDUCATION	Master of Science in Computer Science University of Windsor, Windsor, Ontario 2018 - 2019
	Bachelor of Technology in Computer Science Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Telangana, India 2013 - 2017